(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 989 984**

(51) Int. Cl.:

H04N 19/30 (2014.01)
H04N 19/70 (2014.01)
H04N 19/463 (2014.01)
H04N 19/573 (2014.01)
H04N 19/172 (2014.01)
H04N 19/174 (2014.01)
H04N 19/187 (2014.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **06.10.2020 PCT/US2020/054452**

(87) Fecha y número de publicación internacional: **04.02.2021 WO21022271**

(96) Fecha de presentación y número de la solicitud europea: **06.10.2020 E 20847068 (2)**

(97) Fecha y número de publicación de la concesión europea: **31.07.2024 EP 4032293**

(54) Título: **Evitación de errores en la extracción de subflujo de bits**

(30) Prioridad:

07.10.2019 US 201962911808 P

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
28.11.2024

(73) Titular/es:

HUAWEI TECHNOLOGIES CO., LTD. (100.0%)
Huawei Administration Building, Bantian, Longgang District
Shenzhen, Guangdong 518129, CN

(72) Inventor/es:

WANG, YE-KUI

(74) Agente/Representante:

PONS ARIÑO, Ángel

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

DESCRIPCIÓN

Evitación de errores en la extracción de subflujo de bits

Campo técnico

La presente divulgación se refiere, en general, a la codificación de vídeo, y se refiere, específicamente, a mecanismos para prevenir errores cuando la extracción de subflujos de bits se realiza en un flujo de bits multicapa.

Antecedentes

La cantidad de datos de vídeo necesarios para representar incluso un vídeo relativamente corto puede ser sustancial, lo que puede dar lugar a dificultades cuando los datos se van a transmitir o comunicar de otro modo a través de una red de comunicaciones con capacidad de ancho de banda limitada. Por lo tanto, los datos de vídeo generalmente se comprimen antes de comunicarse a través de las redes de telecomunicaciones modernas. El tamaño de un vídeo también podría ser un problema cuando el vídeo se almacena en un dispositivo de almacenamiento debido a que los recursos de memoria pueden ser limitados. Los dispositivos de compresión de vídeo a menudo usan software y/o hardware en la fuente para la codificación de los datos de vídeo antes de la transmisión o el almacenamiento, lo que reduce la cantidad de datos necesarios para representar imágenes de vídeo digital. Luego, los datos comprimidos son recibidos en el destino por un dispositivo de descompresión de vídeo que decodifica los datos de vídeo. Con recursos de red limitados y demandas cada vez mayores de mayor calidad de vídeo, son deseables técnicas mejoradas de compresión y descompresión que mejoren la relación de compresión con poco o ningún sacrificio en la calidad de la imagen.

El documento WO 2021/057869 A1 (HUAWEI Incorporated) divulga un indicador sps_video_parameter_set_id para indicar si SPS hace referencia a VPS y un indicador GeneralLayerIdx[nuh_layer_id] para indicar que la capa actual es la capa de orden 0.

Document Versatile Video Coding (borrador 6)", 15. El documento de la REUNIÓN JVET; 3 al 12 de julio de 2019; GOTEMBURGO; (THE JOINT VIDEO EXPLORATION TEAM OF ISO/IEC JTC1/SC29/WG11 AND ITU-T SG.16), no. JVET-02001-vE, 31 de julio de 2019, páginas 1-455, XP030293944 divulga el orden de las unidades de NAL y las imágenes codificadas y su asociación con las unidades de acceso a capas y las unidades de evaluación.

El documento AHG8/AHG17: Removing dependencies on VP from the decoding process of a non-scalable bitstream", 16. El documento de la REUNIÓN JVET; 1-11 de octubre de 2019; GINEBRA; (THE JOINT VIDEO EXPLORATION TEAM OF ISO/IEC JTC1/SC29/WG11 AND ITU-T SG.16), no. JVET-P0097, 24 de septiembre de 2019, XP030216235 divulga la eliminación de dependencias en VPS del proceso de decodificación de un flujo de bits no escalable.

Compendio

Las realizaciones de la presente invención están definidas por las reivindicaciones independientes. En las reivindicaciones dependientes se presentan características adicionales de la invención. A continuación, las partes de la descripción y los dibujos que se refieren a realizaciones anteriores que no necesariamente comprenden todas las características para implementar realizaciones de la invención reivindicada no se representan como realizaciones de la invención, sino como ejemplos útiles para comprender las realizaciones de la invención.

Algunos sistemas de codificación de vídeo codifican secuencias de vídeo en capas de imágenes. Las imágenes en diferentes capas tienen diferentes características. Por lo tanto, un codificador puede transmitir diferentes capas a un decodificador dependiendo de las limitaciones del lado del decodificador. Para realizar esta función, un codificador puede codificar todas las capas en un solo flujo de bits. Cuando se le solicita, el codificador puede realizar un proceso de extracción de subflujo de bits para eliminar información superflua del flujo de bits. Esto resulta en un flujo de bits extraído que contiene solo los datos en la una o varias capas, solicitados por el decodificador. Se puede incluir una descripción de cómo se relacionan las capas en un conjunto de parámetros de vídeo (Video Parameter Set, VPS). Una capa de transmisión simultánea es una capa que está configurada para su visualización sin referencia a otras capas. Cuando una capa de transmisión simultánea se transmite a un decodificador, el proceso de extracción de subflujo de bits puede eliminar el VPS, ya que las relaciones de capa no son necesarias para decodificar una capa de transmisión simultánea. Desgraciadamente, ciertas variables en otros conjuntos de parámetros pueden hacer referencia al VPS. Por lo tanto, eliminar el VPS cuando se transmiten capas de transmisión simultánea puede aumentar la eficiencia de la codificación, pero también puede generar errores. El presente ejemplo incluye un mecanismo de codificación de un SPS de una manera que evita errores cuando se elimina un VPS de un flujo de bits codificado como parte de un proceso de extracción de subflujo de bits. El SPS contiene un sps_video_parameter_set_id. El

sps_video_parameter_set_id indica un identificador del VPS que contiene relaciones de capa para la secuencia de vídeo. En un ejemplo, el sps_video_parameter_set_id se ajusta a cero cuando se elimina el VPS antes de la transmisión de un flujo de bits que contiene solo una capa de transmisión simultánea. En otro ejemplo, los SPS utilizados por las capas de transmisión simultánea pueden contener un sps_video_parameter_set_id que se ajusta a cero en el momento de la codificación. En cualquier caso, cuando el sps_video_parameter_set_id se ajusta a cero, las variables relacionadas con SPS que hacen referencia al VPS se ajustan en valores predeterminados para evitar errores. Por ejemplo, un GeneralLayerIdx[nuh_layer_id] indica un índice de capa actual para una capa correspondiente (por ejemplo, la capa de transmisión simultánea). El GeneralLayerIdx[nuh_layer_id] se ajusta a, o se infiere que es cero, cuando el sps_video_parameter_set_id es cero. Como ejemplo adicional, un indicador de capa independiente de VPS para el GeneralLayerIdx[nuh_layer_id] (vps_independent_layer_flag[GeneralLayerIdx[nuh_layer_id]]) se almacena en el VPS y especifica si una capa con índice GeneralLayerIdx[nuh_layer_id] utiliza predicción entre capas. La predicción entre capas no se utiliza para capas de transmisión simultánea. Por lo tanto, el vps_independent_layer_flag[GeneralLayerIdx[nuh_layer_id]] se ajusta a, o se infiere que es, uno, para indicar que no hay predicción entre capas cuando el sps_video_parameter_set_id está ajustado a cero. De esta manera, se evitan errores cuando se elimina un VPS de un flujo de bits antes de la transmisión de una capa de transmisión simultánea. Como resultado, se incrementa la funcionalidad del codificador y del decodificador. Además, la eficiencia de la codificación aumenta al eliminar con éxito un VPS innecesario de un flujo de bits que incluye solo una capa de transmisión simultánea, lo que reduce el uso de recursos de procesador, memoria y/o señalización de red, tanto en el codificador como en el decodificador.

Estas y otras características se comprenderán más claramente a partir de la siguiente descripción detallada tomada junto con los dibujos y reivindicaciones que la acompañan.

Breve descripción de los dibujos

Para una comprensión más completa de esta divulgación se hace referencia a continuación a la siguiente descripción breve, tomada junto con los dibujos adjuntos y la descripción detallada, en donde números de referencia iguales representan partes iguales.

La figura 1 es un diagrama de flujo de un método de ejemplo de codificación de una señal de vídeo.

La figura 2 es un diagrama esquemático de un sistema de codificación y decodificación (códec) de ejemplo para codificación de vídeo.

La figura 3 es un diagrama esquemático que ilustra un codificador de vídeo de ejemplo.

La figura 4 es un diagrama esquemático que ilustra un decodificador de vídeo de ejemplo.

La figura 5 es un diagrama esquemático que ilustra un decodificador de referencia hipotético (Hypothetical Reference Decoder, HRD) de ejemplo.

La figura 6 es un diagrama esquemático que ilustra una secuencia de vídeo multicapa de ejemplo configurada para predicción entre capas.

La figura 7 es un diagrama esquemático que ilustra un flujo de bits de ejemplo.

La figura 8 es un diagrama esquemático de un dispositivo de codificación de vídeo de ejemplo.

La figura 9 es un diagrama de flujo de un método de ejemplo de codificación de una secuencia de vídeo multicapa en un flujo de bits para soportar la eliminación del conjunto de parámetros de vídeo (VPS) durante la extracción del subflujo de bits para capas de transmisión simultánea.

La figura 10 es un diagrama de flujo de un método de ejemplo para decodificar una secuencia de vídeo de un flujo de bits que incluye una capa de transmisión simultánea extraída de un flujo de bits multicapa donde se ha eliminado un VPS durante la extracción del subflujo de bits.

La figura 11 es un diagrama esquemático de un sistema de ejemplo para codificar una secuencia de vídeo multicapa en un flujo de bits para soportar la eliminación de VPS durante la extracción del subflujo de bits para capas de transmisión simultánea.

Descripción detallada

Debe comprenderse desde el principio que, aunque a continuación se proporciona una realización ilustrativa de una o más realizaciones, los sistemas y/o procedimientos descritos pueden implementarse usando cualquier número de técnicas, ya sean conocidas actualmente o existentes. La divulgación no debe de ninguna manera

estar limitada a las implementaciones ilustrativas, dibujos, y técnicas ilustradas a continuación, que incluyen los diseños e implementaciones ejemplares ilustrados y descritos en la presente memoria, sino que pueden modificarse dentro del alcance de las reivindicaciones adjuntas.

La invención se divulga en las figuras 7, 9 y 10 y en los pasajes correspondientes de la descripción. Las otras partes de la descripción, a menos que proporcionen más detalles sobre las características de las reivindicaciones, tienen como objetivo definir el dominio técnico de la invención o proporcionar ejemplos.

Los siguientes términos se definen de la siguiente manera a menos que se utilicen en un contexto contrario en la presente memoria. Específicamente, las siguientes definiciones pretenden proporcionar claridad adicional a la presente divulgación. Sin embargo, los términos pueden describirse de manera diferente en diferentes contextos. En consecuencia, las siguientes definiciones deben ser consideradas como un complemento y no deben ser consideradas como limitantes de ninguna otra definición de las descripciones proporcionadas para dichos términos en la presente memoria.

Un flujo de bits es una secuencia de bits que incluye datos de vídeo que se comprimen para su transmisión entre un codificador y un decodificador. Un codificador es un dispositivo que está configurado para emplear procesos de codificación para comprimir datos de vídeo en un flujo de bits. Un decodificador es un dispositivo que está configurado para emplear procesos de decodificación para reconstruir datos de vídeo a partir de un flujo de bits, para su visualización. Una imagen es una matriz de muestras de luma y/o una matriz de muestras de croma que crean un fotograma o un campo del mismo. Una imagen que se está codificando o decodificando puede denominarse imagen actual para mayor claridad de la explicación. Una imagen codificada es una representación codificada de una imagen que comprende unidades de capa de abstracción de red (Network Abstraction Layer, NAL) de capa de codificación de vídeo (Video Coding Layer, VCL) con un valor particular de identificador de capa de cabecera de unidad de NAL (nuh_layer_id) dentro de una unidad de acceso (Access Unit, AU) y que contiene todas las unidades del árbol de codificación (Coding Tree Unit, CTU) de la imagen. Una imagen decodificada es una imagen producida aplicando un proceso de decodificación a una imagen codificada. Una unidad de NAL es una estructura de sintaxis que contiene datos en forma de carga útil de secuencia de bytes sin procesar (Raw Byte Sequence Payload, RBSP), una indicación del tipo de datos, e intercalados según se desee con bytes de prevención de emulación. Una unidad de NAL de VCL es una unidad de NAL codificada para contener datos de vídeo, tal como un segmento codificado de una imagen. Una unidad de NAL no de VCL es una unidad de NAL que contiene datos que no son de vídeo, tales como sintaxis y/o parámetros que soportan la decodificación de datos de vídeo, la realización de comprobaciones de conformidad u otras operaciones. Una capa es un conjunto de unidades de NAL de VCL que comparten una característica específica (por ejemplo, una resolución, velocidad de fotogramas, tamaño de imagen, etc. comunes) como lo indica el ID (identificador) de capa y las unidades de NAL no de VCL asociadas. Un identificador de capa de cabecera de unidad de NAL (nuh_layer_id) es un elemento de sintaxis que especifica un identificador de una capa que incluye una unidad de NAL.

Un decodificador de referencia hipotético (Hypothetical Reference Decoder, HRD) es un modelo de decodificador que actúa sobre un codificador que comprueba la variabilidad de los flujos de bits producidos mediante un proceso de codificación, para verificar la conformidad con restricciones especificadas. Una prueba de conformidad de flujo de bits es una prueba para determinar si un flujo de bits codificado cumple con un estándar, tal como la codificación de vídeo versátil (Versatile Video Coding, VVC). Un conjunto de parámetros de vídeo (VPS) es una estructura de sintaxis que contiene parámetros relacionados con un vídeo completo. Un conjunto de parámetros de secuencia (Sequence Parameter Set, SPS) es una estructura de sintaxis que contiene elementos de sintaxis que se aplican a cero o más secuencias de vídeo de capa codificadas (Coded Layer Video Sequences, CLVS) completas. Un identificador de conjunto de parámetros de vídeo de SPS (sps_video_parameter_set_id) es un elemento de sintaxis que especifica un identificador (ID) de una referencia de VPS por parte de un SPS. Un índice de capa general (GeneralLayerIdx[i]) es una variable derivada que especifica un índice de una capa i correspondiente. Por lo tanto, una capa actual con un ID de capa de la capa de nuh tiene un índice especificado por GeneralLayerIdx[nuh_layer_id]. Un índice de capa actual es un índice de capa correspondiente a una capa que se está codificando o decodificando. Un indicador de capa independiente de VPS (vps_independent_layer_flag[i]) es un elemento de sintaxis que especifica si una capa i correspondiente utiliza predicción entre capas. Por lo tanto, el vps_independent_layer_flag [GeneralLayerIdx[nuh_layer_id]] especifica si una capa actual utiliza predicción entre capas. La predicción entre capas es un mecanismo de codificación de bloques de valores de muestra en una imagen actual en una capa actual basándose en una o varias imágenes de referencia de una capa diferente (por ejemplo, y en la misma unidad de acceso). Una unidad de acceso (AU) es un conjunto de imágenes codificadas en diferentes capas que están todas asociadas con el mismo tiempo de salida. Un identificador de conjunto de parámetros de VPS (vps_video_parameter_set_id) es un elemento de sintaxis que proporciona un ID para un VPS como referencia para otros elementos/estructuras de sintaxis. Una secuencia de vídeo codificada es un conjunto de una o más imágenes codificadas. Una secuencia de vídeo decodificada es un conjunto de una o más imágenes decodificadas.

En la presente memoria se utilizan las siguientes siglas: Unidad de acceso (AU), Bloque de árbol de codificación

(Coding Tree Block, CTB), Unidad de árbol de codificación (CTU), Unidad de codificación (Coding Unit, CU), Secuencia de vídeo de capa codificada (CLVS), Inicio de secuencia de vídeo de capa codificada (Coded Layer Video Sequence Start, CLVSS), Secuencia de vídeo codificada (Coded Video Sequence, CVS), Inicio de secuencia de vídeo codificada (Coded Video Sequence Start, CVSS), Equipo conjunto de expertos en vídeo (Joint Video Experts Team, JVET), Decodificador de referencia hipotético (HRD), Conjunto de mosaicos con restricción de movimiento (Motion Constrained Tile Set, MCTS), Unidad de transferencia máxima (Maximum Transfer Unit, MTU), Capa de abstracción de red (NAL), Conjunto de capas de salida (Output Layer Set, OLS), Punto de operación (Operation Point, OP), Recuento de orden de imágenes (Picture Order Count, POC), punto de acceso aleatorio (Random Access Point, RAP), Carga útil de secuencia de bytes sin procesar (RBSP), conjunto de parámetros de secuencia (SPS), Conjunto de parámetros de vídeo (VPS), codificación de vídeo versátil (VVC).

Se pueden emplear muchas técnicas de compresión de vídeo para reducir el tamaño de los archivos de vídeo con una pérdida mínima de datos. Por ejemplo, las técnicas de compresión de vídeo pueden incluir la realización de predicciones espaciales (por ejemplo, dentro de una imagen) y/o predicciones temporales (por ejemplo, entre imágenes) para reducir o eliminar la redundancia de datos en secuencias de vídeo. Para la codificación de vídeo basada en bloques, un segmento de vídeo (por ejemplo, una imagen de vídeo o una parte de una imagen de vídeo) se puede dividir en bloques de vídeo, que también pueden denominarse bloques de árbol, bloques de árbol de codificación (CTB), unidades de árbol de codificación (CTU), unidades de codificación (CU) y/o nodos de codificación. Los bloques de vídeo en un segmento intracodificado (I) de una imagen se codifican usando predicción espacial con respecto a muestras de referencia en bloques vecinos en la misma imagen. Los bloques de vídeo en un segmento de predicción (P) unidireccional o predicción bidireccional (B) intercodificado de una imagen se pueden codificar empleando predicción espacial con respecto a muestras de referencia en bloques vecinos en la misma imagen, o predicción temporal con respecto a muestras de referencia en otras imágenes de referencia. Las imágenes pueden denominarse fotogramas y/o imágenes, y las imágenes de referencia pueden denominarse fotogramas de referencia y/o imágenes de referencia. La predicción espacial o temporal da como resultado un bloque predictivo que representa un bloque de imágenes. Los datos residuales representan diferencias de píxel entre el bloque de imágenes original y el bloque predictivo. En consecuencia, un bloque intercodificado se codifica según un vector de movimiento que apunta a un bloque de muestras de referencia que forman el bloque predictivo, y los datos residuales indican la diferencia entre el bloque codificado y el bloque predictivo. Un bloque intracodificado se codifica de acuerdo con un modo de intracodificación y con los datos residuales. Para una mayor compresión, los datos residuales pueden transformarse del dominio de píxel a un dominio de transformación. Estos dan como resultado coeficientes de transformación residuales, que pueden cuantificarse. Los coeficientes de transformación cuantificados pueden disponerse inicialmente en una matriz bidimensional. Los coeficientes de transformación cuantificados se pueden escanear para producir un vector unidimensional de coeficientes de transformación. Se puede aplicar codificación entrópica para lograr una compresión aún mayor. Estas técnicas de compresión de vídeo se explican con mayor detalle a continuación.

Para garantizar que un vídeo codificado se pueda decodificar con precisión, el vídeo se codifica y decodifica de acuerdo con los estándares de codificación de vídeo correspondientes. Los estándares de codificación de vídeo incluyen el segmento de Normalización de la Unión Internacional de Telecomunicaciones (ITU) (ITU-T) H.261, el Grupo de expertos en películas cinematográficas (MPEG)-1 Parte 2 de la Organización Internacional de Normalización/Comisión Electrotécnica Internacional (ISO/IEC), ITU-T H.262 o ISO/IEC MPEG-2 Parte 2, ITU-T H.263, ISO/IEC MPEG-4 Parte 2, Codificación de vídeo avanzada (Advanced Video Coding, AVC), también conocida como ITU-T H.264 o ISO/IEC MPEG-4 Parte 10, y Codificación de vídeo de alta eficiencia (High Efficiency Video Coding, HEVC), también conocida como ITU-T H.265 o MPEG-H Parte 2. AVC incluye extensiones como Codificación de vídeo escalable (Scalable Video Coding, SVC), Codificación de vídeo multivista (Multiview Video Coding, MVC) y Codificación de vídeo multivista más profundidad (Multiview Video Coding plus Depth, MVC+D), y AVC tridimensional (3D) (3D-AVC). HEVC incluye extensiones tales como HEVC escalable (Scalable HEVC, SHVC), HEVC multivista (Multiview HEVC, MV-HEVC) y HEVC 3D (3D-HEVC). El equipo conjunto de expertos en vídeo (JVET) de la ITU-T e ISO/IEC ha comenzado a desarrollar un estándar de codificación de vídeo denominado Codificación de Vídeo Versátil (VVC). VVC está incluido en un Borrador de trabajo (Working Draft, WD), que incluye JVET-02001-v14.

Algunos sistemas de codificación de vídeo codifican secuencias de vídeo en capas de imágenes. Las imágenes en diferentes capas tienen diferentes características. Por lo tanto, un codificador puede transmitir diferentes capas a un decodificador dependiendo de las limitaciones del lado del decodificador. Para realizar esta función, un codificador puede codificar todas las capas en un solo flujo de bits. Cuando se le solicita, el codificador puede realizar un proceso de extracción de subflujo de bits para eliminar información superflua del flujo de bits. Esto resulta en un flujo de bits extraído que contiene solo los datos en la una o varias capas, solicitados por el decodificador. Se puede incluir una descripción de cómo se relacionan las capas en un conjunto de parámetros de vídeo (Video Parameter Set, VPS). Una capa de transmisión simultánea es una capa que está configurada para su visualización sin referencia a otras capas. Cuando una capa de transmisión simultánea se transmite a un decodificador, el proceso de extracción de subflujo de bits puede eliminar el VPS, ya que las relaciones de capa no son necesarias para decodificar una capa de transmisión simultánea. Desgraciadamente, ciertas

variables en otros conjuntos de parámetros pueden hacer referencia al VPS. Por lo tanto, eliminar el VPS cuando se transmiten capas de transmisión simultánea puede aumentar la eficiencia de la codificación, pero también puede generar errores.

en la presente memoria se divulga un mecanismo de codificación de un conjunto de parámetros de secuencia (SPS) de una manera que evita errores cuando se elimina un VPS de un flujo de bits codificado, como parte de un proceso de extracción de subflujo de bits. El SPS contiene un identificador de VPS de SPS (sps_video_parameter_set_id). El sps_video_parameter_set_id indica un identificador del VPS que contiene relaciones de capa para la secuencia de vídeo. En un ejemplo, el sps_video_parameter_set_id se ajusta a cero cuando se elimina el VPS antes de la transmisión de un flujo de bits que contiene solo una capa de transmisión simultánea. En otro ejemplo, los SPS utilizados por las capas de transmisión simultánea pueden contener un sps_video_parameter_set_id que se ajusta a cero en el momento de la codificación. En cualquier caso, cuando el sps_video_parameter_set_id se ajusta a cero, las variables relacionadas con SPS que hacen referencia al VPS se ajustan en valores predeterminados para evitar errores. Por ejemplo, un índice de capa general correspondiente a un identificador de capa de cabecera de unidad de capa de abstracción de red (NAL) (ID de capa de nuh) (GeneralLayerIdx[nuh_layer_id]) indica un índice de capa actual para una capa correspondiente (por ejemplo, la capa de transmisión simultánea). El GeneralLayerIdx[nuh_layer_id] se ajusta a, o se infiere que es, cero cuando el sps_video_parameter_set_id es cero. Como ejemplo adicional, un indicador de capa independiente de VPS para el GeneralLayerIdx[nuh_layer_id] (vps_independent_layer_flag[GeneralLayerIdx[nuh_layer_id]]) se almacena en el VPS y especifica si una capa con índice GeneralLayerIdx[nuh_layer_id] utiliza predicción entre capas. La predicción entre capas no se utiliza para capas de transmisión simultánea. Por lo tanto, el vps_independent_layer_flag[GeneralLayerIdx[nuh_layer_id]] se ajusta a, o se infiere que es, uno, para indicar que no hay predicción entre capas cuando el sps_video_parameter_set_id está ajustado a cero. De esta manera, se evitan errores cuando se elimina un VPS de un flujo de bits antes de la transmisión de una capa de transmisión simultánea. Como resultado, se incrementa la funcionalidad del codificador y del decodificador. Además, la eficiencia de la codificación aumenta al eliminar con éxito un VPS innecesario de un flujo de bits que incluye solo una capa de transmisión simultánea, lo que reduce el uso de recursos de procesador, memoria y/o señalización de red, tanto en el codificador como en el decodificador.

La figura 1 es un diagrama de flujo de un método operativo 100 de ejemplo, de codificación de una señal de vídeo. Específicamente, una señal de vídeo se codifica en un codificador. El proceso de codificación comprime la señal de vídeo empleando diversos mecanismos para reducir el tamaño del archivo de vídeo. Un tamaño de archivo más pequeño permite que el archivo de vídeo comprimido se transmita a un usuario, al tiempo que se reduce la sobrecarga de ancho de banda asociada. Luego, el decodificador decodifica el archivo de vídeo comprimido para reconstruir la señal de vídeo original para su visualización por un usuario final. El proceso de decodificación refleja de manera general el proceso de codificación para permitir que el decodificador reconstruya coherentemente la señal de vídeo.

En la etapa 101, la señal de vídeo se introduce en el codificador. Por ejemplo, la señal de vídeo puede ser un archivo de vídeo sin comprimir almacenado en la memoria. Como ejemplo adicional, el archivo de vídeo puede capturarse mediante un dispositivo de captura de vídeo, tal como una cámara de vídeo, y ser codificado para soportar la transmisión en continuo del vídeo. El archivo de vídeo puede incluir tanto un componente de audio como un componente de vídeo. El componente de vídeo contiene una serie de fotogramas de imágenes que, cuando se ven en secuencia, dan la impresión visual de movimiento. Los fotogramas contienen píxeles que se expresan en términos de luz, denominados en la presente memoria componentes de luma (o muestras de luma), y color, denominados componentes de croma (o muestras de color). En algunos ejemplos, los fotogramas también pueden contener valores de profundidad para soportar la visualización tridimensional.

En la etapa 103, el vídeo se divide en bloques. La división incluye subdividir los píxeles de cada fotograma en bloques cuadrados y/o rectangulares para su compresión. Por ejemplo, en la codificación de vídeo de alta eficiencia (HEVC) (también conocida como H.265 y MPEG-H Parte 2), el fotograma se puede dividir primero en unidades de árbol de codificación (CTU), que son bloques de un tamaño predefinido (por ejemplo, sesenta y cuatro píxeles por sesenta y cuatro píxeles). Las CTU contienen muestras de luma y croma. Se pueden emplear árboles de codificación para dividir las CTU en bloques y luego subdividir recursivamente los bloques hasta que se consigan configuraciones que soporten codificación adicional. Por ejemplo, los componentes de luma de un fotograma pueden subdividirse hasta que los bloques individuales contengan valores de iluminación relativamente homogéneos. Además, los componentes de croma de un fotograma pueden subdividirse hasta que los bloques individuales contengan valores de color relativamente homogéneos. En consecuencia, los mecanismos de división varían dependiendo del contenido de los fotogramas de vídeo.

En la etapa 105, se emplean diversos mecanismos de compresión para comprimir los bloques de imágenes divididos en la etapa 103. Por ejemplo, se puede emplear interpredicción y/o intrapredicción. La interpredicción está diseñada para aprovechar el hecho de que los objetos en una escena común tienden a aparecer en fotogramas sucesivos. En consecuencia, no es necesario describir repetidamente un bloque que representa un objeto en un fotograma de referencia, en fotogramas adyacentes. Específicamente, un objeto, tal como una

mesa, puede permanecer en una posición constante en varios fotogramas. Por lo tanto, la mesa se describe una vez y los fotogramas adyacentes pueden hacer referencia al fotograma de referencia. Se pueden emplear mecanismos de coincidencia de patrones para hacer coincidir objetos en múltiples fotogramas. Además, los objetos en movimiento pueden representarse en múltiples fotogramas, por ejemplo debido al movimiento del objeto o al movimiento de la cámara. Como ejemplo particular, un vídeo puede mostrar un automóvil que se mueve por la pantalla en varios fotogramas. Se pueden emplear vectores de movimiento para describir dicho movimiento. Un vector de movimiento es un vector bidimensional que proporciona un desplazamiento desde las coordenadas de un objeto en un fotograma hasta las coordenadas del objeto en un fotograma de referencia. Por lo tanto, la interpredicción puede codificar un bloque de imágenes en un fotograma actual como un conjunto de vectores de movimiento que indican un desplazamiento de un bloque correspondiente en un fotograma de referencia.

La intrapredicción codifica bloques en un fotograma común. La intrapredicción aprovecha el hecho de que los componentes de luma y croma tienden a agruparse en un fotograma. Por ejemplo, una mancha de verde en una parte de un árbol tiende a ubicarse adyacente a manchas de verde similares. La intrapredicción emplea múltiples modos de predicción direccional (por ejemplo, treinta y tres en HEVC), un modo plano y un modo de corriente continua (DC). Los modos direccionales indican que un bloque actual es similar/igual que las muestras de un bloque vecino en la dirección correspondiente. El modo plano indica que una serie de bloques a lo largo de una fila/columna (por ejemplo, un plano) se puede interpolar en función de los bloques vecinos en los bordes de la fila. El modo plano, en efecto, indica una transición suave de luz/color a través de una fila/columna empleando una pendiente relativamente constante en los valores que cambian. El modo de DC se emplea para suavizar los límites e indica que un bloque es similar/igual que un valor promedio asociado con muestras de todos los bloques vecinos asociados con las direcciones angulares de los modos de predicción direccional. En consecuencia, los bloques de intrapredicción pueden representar bloques de imágenes como diversos valores de modo de predicción relacional, en lugar de los valores reales. Además, los bloques de interpredicción pueden representar bloques de imágenes como valores de vector de movimiento, en lugar de valores reales. En cualquier caso, es posible que los bloques de predicción no representen exactamente los bloques de imagen en algunos casos. Cualquier diferencia se almacena en bloques residuales. Se pueden aplicar transformaciones a los bloques residuales para comprimir aún más el archivo.

En la etapa 107, se pueden aplicar diversas técnicas de filtrado. En HEVC, los filtros se aplican según un esquema de filtrado en bucle. La predicción basada en bloques explicada anteriormente puede dar como resultado la creación de imágenes en bloques en el decodificador. Además, el esquema de predicción basado en bloques puede codificar un bloque y luego reconstruir el bloque codificado para su uso posterior como bloque de referencia. El esquema de filtrado en bucle aplica de manera iterativa filtros de supresión de ruido, filtros de desbloqueo, filtros de bucle adaptativos y filtros de desplazamiento adaptativo de muestra (Sample Adaptive Offset, SAO) a los bloques/fotogramas. Estos filtros mitigan dichos artefactos de bloqueo de modo que el archivo codificado pueda ser reconstruido con precisión. Además, estos filtros mitigan los artefactos en los bloques de referencia reconstruidos, de modo que sea menos probable que los artefactos creen artefactos adicionales en bloques posteriores que se codifican en basándose en los bloques de referencia reconstruidos.

Una vez que la señal de vídeo ha sido dividida, comprimida y filtrada, los datos resultantes se codifican en un flujo de bits en la etapa 109. El flujo de bits incluye los datos explicados anteriormente, así como cualquier dato de señalización deseado para soportar la reconstrucción adecuada de la señal de vídeo en el decodificador. Por ejemplo, dichos datos pueden incluir datos de división, datos de predicción, bloques residuales y diversos indicadores que proporcionan instrucciones de codificación al decodificador. El flujo de bits puede almacenarse en la memoria para su transmisión hacia un decodificador, previa solicitud. El flujo de bits también puede difundirse y/o multidifundirse hacia una pluralidad de decodificadores. La creación del flujo de bits es un proceso iterativo. En consecuencia, las etapas 101, 103, 105, 107 y 109 pueden ocurrir de manera continua y/o simultánea en muchos fotogramas y bloques. El orden mostrado en la figura 1 se presenta para mayor claridad y facilidad de explicación, y no pretende limitar el proceso de codificación de vídeo a un orden particular.

El decodificador recibe el flujo de bits y comienza el proceso de decodificación en la etapa 111. Específicamente, el decodificador emplea un esquema de decodificación entrópica para convertir el flujo de bits en la sintaxis y los datos de vídeo correspondientes. El decodificador emplea los datos de sintaxis del flujo de bits para determinar las divisiones para las tramas en la etapa 111. La división debe coincidir con los resultados de la división del bloque en la etapa 103. A continuación se describe la codificación/decodificación entrópica empleada en la etapa 111. El codificador toma muchas decisiones durante el proceso de compresión, tales como seleccionar esquemas de división de bloques entre varias opciones posibles, basándose en el posicionamiento espacial de los valores en la o las imágenes de entrada. Señalizar las opciones exactas puede emplear una gran cantidad de bins. Tal como se utiliza en la presente memoria, un bin es un valor binario que se trata como una variable (por ejemplo, un valor de bit que puede variar dependiendo del contexto). La codificación entrópica permite al codificador descartar cualquier opción que claramente no sea viable para un caso particular, dejando un conjunto de opciones permitidas. Luego, a cada opción permitida se le asigna una palabra de código. La longitud de las palabras de código se basa en el número de opciones permitidas (por ejemplo, un bin para dos opciones, dos bins para tres o cuatro opciones, etc.). Luego, el codificador codifica la

palabra de código para la opción seleccionada. Este esquema reduce el tamaño de las palabras de código, ya que las palabras de código son tan grandes como se desee para indicar de manera única una selección de un pequeño subconjunto de opciones permitidas, en lugar de indicar de manera única la selección de un conjunto potencialmente grande de todas las opciones posibles. Luego, el decodificador decodifica la selección determinando el conjunto de opciones permitidas, de manera similar, al codificador. Determinando el conjunto de opciones permitidas, el decodificador puede leer la palabra de código y determinar la selección realizada por el codificador.

En la etapa 113, el decodificador realiza la decodificación de bloques. Específicamente, el decodificador emplea transformaciones inversas para generar bloques residuales. Luego, el decodificador emplea los bloques residuales y los bloques de predicción correspondientes para reconstruir los bloques de imágenes de acuerdo con la división. Los bloques de predicción pueden incluir tanto bloques de intrapredicción como bloques de interpredicción generados en el codificador en la etapa 105. Los bloques de imágenes reconstruidos se colocan a continuación en fotogramas de una señal de vídeo reconstruida de acuerdo con los datos de división determinados en la etapa 111. La sintaxis para la etapa 113 también puede señalarse en el flujo de bits mediante codificación entrópica tal como se explicó anteriormente.

En la etapa 115, el filtrado se realiza en los fotogramas de la señal de vídeo reconstruida de una manera similar a la etapa 107 en el codificador. Por ejemplo, se pueden aplicar filtros de supresión de ruido, filtros de desbloqueo, filtros de bucle adaptativo y filtros de SAO a los fotogramas, para eliminar artefactos de bloqueo. Una vez filtrados los fotogramas, la señal de vídeo puede enviarse a una pantalla en la etapa 117 para su visualización por un usuario final.

La figura 2 es un diagrama esquemático de un sistema de codificación y decodificación (códec) 200 de ejemplo, para codificación de vídeo. Específicamente, el sistema de códec 200 proporciona funcionalidad para soportar la implementación del método operativo 100. El sistema de códec 200 está generalizado para representar componentes empleados tanto en un codificador como en un decodificador. El sistema de códec 200 recibe y divide una señal de vídeo como se explica con respecto a las etapas 101 y 103 en el método operativo 100, lo que da como resultado una señal de vídeo dividida 201. A continuación, el sistema de códec 200 comprime la señal de vídeo dividida 201 en un flujo de bits codificado cuando actúa como un codificador, tal como se explica con respecto a las etapas 105, 107 y 109 en el método 100. Cuando actúa como decodificador, el sistema de códec 200 genera una señal de vídeo de salida a partir del flujo de bits, tal como se explica con respecto a las etapas 111, 113, 115 y 117 en el método operativo 100. El sistema de códec 200 incluye un componente de control del codificador general 211, un componente de cuantificación y escalado de transformación 213, un componente de estimación de intra-imagen 215, un componente de predicción de intra-imagen 217, un componente de compensación de movimiento 219, un componente de estimación de movimiento 221, un componente de escalado y transformación inversa 229, un componente de análisis de control de filtro 227, un componente de filtros en bucle 225, un componente de memoria intermedia de imágenes decodificadas 223 y un componente de formateo de cabecera y codificación aritmética binaria adaptativa al contexto (Context Adaptive Binary Arithmetic Coding, CABAC) 231. Dichos componentes están acoplados como se muestra. En la figura 2, las líneas continuas indican el movimiento de datos a codificar/decodificar mientras que las líneas discontinuas indican el movimiento de datos de control que controlan el funcionamiento de otros componentes. Todos los componentes del sistema de códec 200 pueden estar presentes en el codificador. El decodificador puede incluir un subconjunto de los componentes del sistema de códec 200. Por ejemplo, el decodificador puede incluir el componente de predicción de intra-imagen 217, el componente de compensación de movimiento 219, el componente de escalado y transformación inversa 229, el componente de filtros en bucle 225 y el componente de memoria intermedia de imágenes decodificadas 223. Estos componentes se describen a continuación.

La señal de vídeo dividida 201 es una secuencia de vídeo capturada que ha sido dividida en bloques de píxeles mediante un árbol de codificación. Un árbol de codificación emplea diversos modos de división para subdividir un bloque de píxeles en bloques de píxeles más pequeños. Estos bloques se pueden subdividir aún más en bloques más pequeños. Los bloques pueden denominarse nodos en el árbol de codificación. Los nodos principales más grandes se dividen en nodos secundarios más pequeños. El número de veces que se subdivide un nodo se denomina profundidad del nodo/árbol de codificación. Los bloques divididos pueden ser incluidos en unidades de codificación (Coding Units, CU) en algunos casos. Por ejemplo, una CU puede ser una subparte de una CTU que contiene un bloque de luma, uno o más bloques de croma de diferencia roja (Cr) y uno o más bloques de croma de diferencia azul (Cb) junto con las instrucciones de sintaxis correspondientes para la CU. Los modos de división pueden incluir un árbol binario (Binary Tree, BT), un árbol triple (Triple Tree, TT) y un árbol cuádruple (Quad Tree, QT) empleados para dividir un nodo en dos, tres o cuatro nodos secundarios, respectivamente, de diferentes formas según los modos de división empleados. La señal de vídeo dividida 201 se envía al componente de control del codificador general 211, al componente de escalado y cuantificación de transformación 213, al componente de estimación de intra-imagen 215, al componente de análisis de control de filtro 227 y al componente de estimación de movimiento 221, para su compresión.

El componente de control del codificador general 211 está configurado para tomar decisiones relacionadas con

la codificación de las imágenes de la secuencia de vídeo en el flujo de bits según las restricciones de la aplicación. Por ejemplo, el componente de control del codificador general 211 gestiona la optimización de la velocidad de bits/tamaño del flujo de bits frente a la calidad de reconstrucción. Dichas decisiones se pueden tomar basándose en la disponibilidad de espacio de almacenamiento/ancho de banda y en las solicitudes de resolución de imagen. El componente de control del codificador general 211 también gestiona la utilización de la memoria intermedia a la luz de la velocidad de transmisión, para mitigar los problemas de insuficiencia y desbordamiento de la memoria intermedia. Para gestionar estos problemas, el componente de control del codificador general 211 gestiona la división, la predicción y el filtrado por parte de los otros componentes. Por ejemplo, el componente de control del codificador general 211 puede aumentar dinámicamente la complejidad de la compresión, para aumentar la resolución y aumentar el uso de ancho de banda, o disminuir la complejidad de la compresión, para disminuir la resolución y el uso de ancho de banda. Por lo tanto, el componente de control del codificador general 211 controla los otros componentes del sistema de códec 200 para equilibrar la calidad de reconstrucción de la señal de vídeo con las cuestiones de la velocidad de bits. El componente de control del codificador general 211 crea datos de control, que controlan el funcionamiento de los otros componentes. Los datos de control también se envían al componente de formateo de cabecera y CABAC 231 para ser codificados en el flujo de bits para indicar parámetros para su decodificación en el decodificador.

La señal de vídeo dividida 201 también se envía al componente de estimación de movimiento 221 y al componente de compensación de movimiento 219, para interpredicción. Un fotograma o segmento de la señal de vídeo dividida 201 se puede dividir en múltiples bloques de vídeo. El componente de estimación de movimiento 221 y el componente de compensación de movimiento 219 realizan una codificación interpredictiva del bloque de vídeo recibido en relación con uno o más bloques en uno o más fotogramas de referencia, para proporcionar una predicción temporal. El codificador 200 de vídeo puede realizar varias pasadas de codificación, por ejemplo, para seleccionar un modo de codificación apropiado para cada bloque de datos de vídeo.

La unidad de estimación de movimiento 221 y la unidad de compensación de movimiento 219 pueden estar altamente integradas, pero se ilustran por separado con fines conceptuales. La estimación de movimiento, realizada por la unidad de estimación de movimiento 221, es el proceso de generar vectores de movimiento que estiman el movimiento para bloques de vídeo. Un vector de movimiento, por ejemplo, puede indicar el desplazamiento de un objeto codificado con respecto a un bloque predictivo. Un bloque predictivo es un bloque que coincide estrechamente con el bloque que se va a codificar, en términos de diferencia de píxel. Un bloque predictivo también puede denominarse bloque de referencia. Dicha diferencia de píxel puede determinarse mediante la suma de la diferencia absoluta (Sum of Absolute Difference, SAD), la suma de la diferencia cuadrática (Sum of Square Difference, SSD) u otras métricas de diferencia. HEVC emplea varios objetos codificados, incluida una CTU, bloques de árbol de codificación (CTB) y varias CU. Por ejemplo, una CTU se puede dividir en varios CTB, que luego se pueden dividir en varios CB para su inclusión en las CU. Una CU puede codificarse como una unidad de predicción (Prediction Unit, PU) que contiene datos de predicción y/o una unidad de transformación (Transform Unit, TU) que contiene datos residuales transformados para la CU. El componente de estimación de movimiento 221 genera vectores de movimiento, PU y TU utilizando un análisis de distorsión de velocidad como parte de un proceso de optimización de distorsión de velocidad. Por ejemplo, el componente de estimación de movimiento 221 puede determinar múltiples bloques de referencia, múltiples vectores de movimiento, etc. para un bloque/fotograma actual, y puede seleccionar los bloques de referencia, vectores de movimiento, etc. que tengan las mejores características de distorsión de velocidad. Las mejores características de distorsión de velocidad equilibran tanto la calidad de la reconstrucción de vídeo (por ejemplo, la cantidad de datos perdidos por compresión) con la eficiencia de codificación (por ejemplo, el tamaño de la codificación final).

En algunos ejemplos, el sistema de códec 200 puede calcular valores para posiciones de píxel sub-enteras de imágenes de referencia almacenadas en el componente de memoria intermedia de imágenes decodificadas 223. Por ejemplo, el codificador 200 de vídeo puede interpolar valores de posiciones de un cuarto de píxel, posiciones de un octavo de píxel u otras posiciones fraccionarias de píxel, de la imagen de referencia. Por lo tanto, la unidad de estimación de movimiento 221 puede realizar una búsqueda de movimiento con respecto a posiciones de píxel completas y posiciones de píxel fraccionarias, y generar un vector de movimiento con precisión de píxel fraccionaria. El componente de estimación de movimiento 221 calcula un vector de movimiento para una PU de un bloque de vídeo en un segmento intercodificado comparando la posición de la PU con la posición de un bloque predictivo de una imagen de referencia. El componente de estimación de movimiento 221 envía el vector de movimiento calculado como datos de movimiento al componente de formateo de cabecera y CABAC 231 para codificación y movimiento, al componente de compensación de movimiento 219.

La compensación de movimiento, realizada por el componente de compensación de movimiento 219, puede implicar la obtención o generación del bloque predictivo basándose en el vector de movimiento determinado por el componente de estimación de movimiento 221. De nuevo, el componente de estimación de movimiento 221 y el componente de compensación de movimiento 219 pueden estar integrados funcionalmente, en algunos ejemplos. Tras la recepción del vector de movimiento para la PU del bloque de vídeo actual, el

componente de compensación de movimiento 219 puede ubicar el bloque predictivo al que apunta el vector de movimiento. A continuación, se forma un bloque de vídeo residual restando los valores de píxeles del bloque predictivo de los valores de píxel del bloque de vídeo actual que se está codificando, formando valores de diferencia de píxel. En general, el componente de estimación de movimiento 221 realiza una estimación de movimiento relativa a los componentes de luma, y el componente de compensación de movimiento 219 utiliza vectores de movimiento calculados basándose en los componentes de luma tanto para los componentes de croma como para los componentes de luma. El bloque predictivo y el bloque residual se reenvían al componente de escalado y cuantificación de transformación 213.

La señal de vídeo dividida 201 también se envía al componente de estimación de intra-imagen 215 y al componente de predicción de intra-imagen 217. Al igual que con el componente de estimación de movimiento 221 y el componente de compensación de movimiento 219, el componente de estimación de intra-imagen 215 y el componente de predicción de intra-imagen 217 pueden estar altamente integrados, pero se ilustran por separado con fines conceptuales. El componente de estimación de intra-imagen 215 y el componente de predicción de intra-imagen 217 intrapredicen un bloque actual con respecto a los bloques en un fotograma actual, como alternativa a la interpredicción realizada por el componente de estimación de movimiento 221 y el componente de compensación de movimiento 219 entre fotogramas, tal como se describió anteriormente. En particular, la unidad de intrapredicción 215 puede determinar un modo de intrapredicción para ser utilizado para codificar un bloque actual. En algunos ejemplos, el componente de estimación de intra-imagen 215 selecciona un modo de intrapredicción apropiado de codificación de un bloque actual a partir de múltiples modos de intrapredicción probados. Los modos de intrapredicción seleccionados se reenvían a continuación al componente de formateo de cabecera y CABAC 231 para su codificación.

Por ejemplo, el componente de estimación de intra-imagen 215 calcula valores de distorsión de velocidad utilizando un análisis de distorsión de velocidad para los diversos modos de intrapredicción probados, y selecciona el modo de intrapredicción que tiene las mejores características de distorsión de velocidad entre los modos probados. El análisis de distorsión de velocidad determina de manera general una cantidad de distorsión (o error) entre un bloque codificado y un bloque original sin codificar que fue codificado para producir el bloque codificado, así como una velocidad de bits (es decir, una cantidad de bits) utilizada para producir el bloque codificado. El componente de estimación de intra-imagen 215 calcula relaciones a partir de las distorsiones y velocidades para los diversos bloques codificados, para determinar qué modo de intrapredicción presenta el mejor valor de distorsión de velocidad para el bloque. Además, el componente de estimación de intra-imagen 215 puede configurarse para la codificación de bloques de profundidad de un mapa de profundidad usando un modo de modelización de profundidad (Depth Modeling Mode, DMM) basado en la optimización de la distorsión de velocidad (Rate-Distortion Optimization, RDO).

El componente de predicción de intra-imagen 217 puede generar un bloque residual a partir del bloque predictivo basándose en los modos de intrapredicción seleccionados determinados por el componente de estimación de intra-imagen 215 cuando está implementado en un codificador, o lee el bloque residual del flujo de bits cuando está implementado en un decodificador. El bloque residual incluye la diferencia de valores entre el bloque predictivo y el bloque original, representado como una matriz. A continuación, el bloque residual es reenviado al componente de escalado y cuantificación de transformación 213. El componente de estimación de intra-imagen 215 y el componente de predicción de intra-imagen 217 pueden funcionar tanto en componentes de luma como de croma.

El componente de cuantificación y escalado de transformación 213 está configurado para comprimir aún más el bloque residual. El componente de escalado y cuantificación de transformación 213 aplica una transformada, tal como una transformada de coseno discreta (Discrete Cosine Transform, DCT), una transformada de seno discreta (Discrete Sine Transform, DST) o una transformada conceptualmente similar, al bloque residual, produciendo un bloque de vídeo que comprende valores de coeficientes de transformación residuales. También podrían usarse transformadas de ondícula, transformadas de enteros, transformadas de subbanda u otros tipos de transformadas. La transformada puede convertir la información residual de un dominio de valor de píxel en un dominio de transformada, tal como un dominio de la frecuencia. El componente de escalado y cuantificación de transformación 213 también está configurado para escalar la información residual transformada, por ejemplo basándose en la frecuencia. Dicho escalado implica aplicar un factor de escala a la información residual de modo que se cuantifique información de frecuencia diferente con diferentes granularidades, lo que puede afectar a la calidad visual final del vídeo reconstruido. El componente de cuantificación y escalado de transformación 213 también está configurado para cuantificar los coeficientes de transformación para reducir aún más la velocidad de bits. El proceso de cuantificación puede reducir la profundidad de bits asociada con algunos o todos los coeficientes. El grado de cuantificación puede modificarse ajustando un parámetro de cuantificación. En algunos ejemplos, el componente de cuantificación y escalado de transformación 213 puede entonces realizar un escaneo de la matriz que incluye los coeficientes de transformación cuantificados. Los coeficientes de transformación cuantificados se envían al componente de formateo de cabecera y CABAC 231 para ser codificados en el flujo de bits.

El componente de escalado y transformación inversa 229 aplica una operación inversa del componente de

escalado y cuantificación de transformación 213 para soportar la estimación de movimiento. El componente de escalado y transformación inversa 229 aplica escalado, transformación y/o cuantificación inversa para reconstruir el bloque residual en el dominio de píxel, por ejemplo, para su uso posterior como bloque de referencia que puede convertirse en un bloque predictivo para otro bloque actual. El componente de estimación de movimiento 221 y/o el componente de compensación de movimiento 219 pueden calcular un bloque de referencia sumando el bloque residual nuevamente a un bloque predictivo correspondiente para su uso en la estimación de movimiento de un bloque/fotograma posterior. Se aplican filtros a los bloques de referencia reconstruidos para mitigar los artefactos creados durante el escalado, la cuantificación y la transformación. De lo contrario, dichos artefactos podrían causar predicciones inexactas (y crear artefactos adicionales) cuando se predicen bloques posteriores.

El componente de análisis de control de filtro 227 y el componente de filtros en bucle 225 aplican los filtros a los bloques residuales y/o a los bloques de imágenes reconstruidos. Por ejemplo, el bloque residual transformado del componente de escalado y transformación inversa 229 se puede combinar con un bloque de predicción correspondiente del componente de predicción de intra-imagen 217 y/o el componente de compensación de movimiento 219, para reconstruir el bloque de imágenes original. Luego, los filtros se pueden aplicar al bloque de imágenes reconstruido. En algunos ejemplos, los filtros pueden aplicarse a los bloques residuales. Como con otros componentes en la figura 2, el componente de análisis de control de filtro 227 y el componente de filtros en bucle 225 están altamente integrados y pueden implementarse juntos, pero se representan por separado con fines conceptuales. Los filtros aplicados a los bloques de referencia reconstruidos se aplican a regiones espaciales particulares e incluyen múltiples parámetros para ajustar cómo se aplican dichos filtros. El componente de análisis de control de filtro 227 analiza los bloques de referencia reconstruidos para determinar dónde se deben aplicar dichos filtros, y ajusta los parámetros correspondientes. Dichos datos se reenvían al componente de formateo de cabecera y CABAC 231 como datos de control de filtro, para su codificación. El componente de filtros en bucle 225 aplica dichos filtros basándose en los datos de control de filtro. Los filtros pueden incluir un filtro de desbloqueo, un filtro de supresión de ruido, un filtro de SAO y un filtro de bucle adaptativo. Dichos filtros se pueden aplicar en el dominio espacial/de píxel (por ejemplo, en un bloque de píxeles reconstruido) o en el dominio de la frecuencia, según el ejemplo.

Cuando funciona como un codificador, el bloque de imágenes reconstruido filtrado, el bloque residual y/o el bloque de predicción se almacenan en el componente de memoria intermedia de imágenes decodificadas 223 para su uso posterior en la estimación del movimiento, tal como se explicó anteriormente. Cuando funciona como decodificador, el componente de memoria intermedia de imágenes decodificadas 223 almacena y reenvía los bloques reconstruidos y filtrados hacia una pantalla, como parte de una señal de vídeo de salida. El componente de memoria intermedia de imágenes decodificadas 223 puede ser cualquier dispositivo de memoria capaz de almacenar bloques de predicción, bloques residuales y/o bloques de imágenes reconstruidos.

El componente de formateo de cabecera y CABAC 231 recibe los datos de los diversos componentes del sistema de códec 200, y codifica dichos datos en un flujo de bits codificado para su transmisión hacia un decodificador. Específicamente, el componente de formateo de cabecera y CABAC 231 genera diversas cabeceras de codificación de datos de control, tales como datos de control general y datos de control de filtro. Además, los datos de predicción, incluidos los datos de intrapredicción y de movimiento, así como los datos residuales en forma de datos de coeficientes de transformación cuantificados, están todos codificados en el flujo de bits. El flujo de bits final incluye toda la información deseada por el decodificador para reconstruir la señal de vídeo dividida 201 original. Dicha información también puede incluir tablas de índice de modos de intrapredicción (también denominadas tablas de mapeo de palabras de código), definiciones de contextos de codificación para diversos bloques, indicaciones de los modos de intrapredicción más probables, una indicación de información de división, etc. Dichos datos pueden ser codificados empleando codificación entrópica. Por ejemplo, la información puede codificarse empleando codificación de longitud variable adaptativa al contexto (Context Adaptive Variable Length Coding, CAVLC), CABAC, codificación aritmética binaria adaptativa al contexto basada en sintaxis (Syntax-based context-adaptive Binary Arithmetic Coding, SBAC), codificación entrópica de partición de intervalo de probabilidad (Probability Interval Partitioning Entropy, PIPE) u otra técnica de codificación entrópica. Después de la codificación entrópica, el flujo de bits codificado puede transmitirse a otro dispositivo (por ejemplo, un decodificador de vídeo) o archivarse para su posterior transmisión o recuperación.

La figura 3 es un diagrama de bloques que ilustra un codificador 300 de vídeo de ejemplo. El codificador 300 de vídeo puede emplearse para implementar las funciones de codificación del sistema de códec 200 y/o implementar las etapas 101, 103, 105, 107 y/o 109 del método operativo 100. El codificador 300 divide una señal de vídeo de entrada, dando como resultado una señal de vídeo dividida 301, que es sustancialmente similar a la señal de vídeo dividida 201. A continuación, la señal de vídeo dividida 301 es comprimida y codificada en un flujo de bits mediante componentes del codificador 300.

Específicamente, la señal de vídeo dividida 301 se reenvía a un componente de predicción de intra-imagen 317 para intrapredicción. El componente de predicción de intra-imagen 317 puede ser sustancialmente similar

al componente de estimación de intra-imagen 215 y al componente de predicción de intra-imagen 217. La señal de vídeo dividida 301 también se envía a un componente de compensación de movimiento 321 para interpredicción basada en bloques de referencia en un componente de memoria intermedia de imágenes decodificadas 323. El componente de compensación de movimiento 321 puede ser sustancialmente similar al componente de estimación de movimiento 221 y al componente de compensación de movimiento 219. Los bloques de predicción y los bloques residuales del componente de predicción de intra-imagen 317 y el componente de compensación de movimiento 321 se envían a un componente de transformación y cuantificación 313 para la transformación y cuantificación de los bloques residuales. El componente de transformación y cuantificación 313 puede ser sustancialmente similar al componente de transformación y cuantificación 213. Los bloques residuales transformados y cuantificados y los bloques de predicción correspondientes (junto con los datos de control asociados) se reenvían a un componente de codificación entrópica 331 para su codificación en un flujo de bits. El componente de codificación entrópica 331 puede ser sustancialmente similar al componente de formateo de cabecera y CABAC 231.

Los bloques residuales transformados y cuantificados y/o los bloques de predicción correspondientes también se reenvían desde el componente de transformación y cuantificación 313 a un componente de transformación y cuantificación inversa 329, para su reconstrucción en bloques de referencia para su uso por el componente de compensación de movimiento. 321. El componente de transformación inversa y cuantificación 329 puede ser sustancialmente similar al componente de escalado y transformación inversa 229. Los filtros en bucle en un componente de filtros en bucle 325 también se aplican a los bloques residuales y/o a los bloques de referencia reconstruidos, dependiendo del ejemplo. El componente de filtros en bucle 325 puede ser sustancialmente similar al componente de análisis de control de filtro 227 y al componente de filtros en bucle 225. El componente de filtros en bucle 325 puede incluir múltiples filtros, tal como se explica con respecto al componente de filtros en bucle 225. A continuación, los bloques filtrados son almacenados en un componente de memoria intermedia de imágenes decodificadas 323 para su uso como bloques de referencia por parte del componente de compensación de movimiento 321. El componente de memoria intermedia de imágenes decodificadas 323 puede ser sustancialmente similar al componente de memoria intermedia de imágenes decodificadas 223.

La figura 4 es un diagrama de bloques que ilustra un decodificador 400 de vídeo de ejemplo. El decodificador 400 de vídeo puede emplearse para implementar las funciones de decodificación del sistema de códec 200 y/o implementar las etapas 111, 113, 115 y/o 117 del método operativo 100. El decodificador 400 recibe un flujo de bits, por ejemplo de un codificador 300, y genera una señal de vídeo de salida reconstruida basada en el flujo de bits, para su visualización por parte de un usuario final.

El flujo de bits es recibido por un componente de decodificación entrópica 433. El componente de decodificación entrópica 433 está configurado para implementar un esquema de decodificación entrópica, tal como codificación CAVLC, CABAC, SBAC, PIPE u otras técnicas de codificación entrópica. Por ejemplo, el componente de decodificación entrópica 433 puede emplear información de cabecera para proporcionar un contexto para interpretar datos adicionales codificados como palabras de código en el flujo de bits. La información decodificada incluye cualquier información deseada para decodificar la señal de vídeo, tal como datos de control general, datos de control de filtro, información de división, datos de movimiento, datos de predicción y coeficientes de transformación cuantificados de bloques residuales. Los coeficientes de transformación cuantificados se envían a un componente de cuantificación y transformación inversa 429 para su reconstrucción en bloques residuales. El componente de transformación inversa y cuantificación 429 puede ser similar al componente de transformación inversa y cuantificación 329.

Los bloques residuales reconstruidos y/o los bloques de predicción se reenvían al componente de predicción de intra-imagen 417 para su reconstrucción en bloques de imágenes basándose en operaciones de intrapredicción. El componente de predicción de intra-imagen 417 puede ser sustancialmente similar al componente de estimación de intra-imagen 215 y al componente de predicción de intra-imagen 217. Específicamente, el componente de predicción de intra-imagen 417 emplea modos de predicción para localizar un bloque de referencia en el fotograma y aplica un bloque residual al resultado para reconstruir bloques de imágenes intrapredichos. Los bloques de imagen intrapredichos reconstruidos y/o los bloques residuales y los datos de interpredicción correspondientes se reenvían a un componente de memoria intermedia de imágenes decodificadas 423 a través de un componente de filtros en bucle 425, que puede ser sustancialmente similar al componente de memoria intermedia de imágenes decodificadas 223 y al componente de filtros en bucle 225, respectivamente. El componente de filtros en bucle 425 filtra los bloques de imágenes reconstruidos, los bloques residuales y/o los bloques de predicción, y dicha información se almacena en el componente de memoria intermedia de imágenes decodificadas 423. Los bloques de imágenes reconstruidos a partir del componente de memoria intermedia de imágenes decodificadas 423 se reenvían a un componente de compensación de movimiento 421 para interpredicción. El componente de compensación de movimiento 421 puede ser sustancialmente similar al componente de estimación de movimiento 221 y/o al componente de compensación de movimiento 219. Específicamente, el componente de compensación de movimiento 421 emplea vectores de movimiento de un bloque de referencia para generar un bloque de predicción, y aplica un bloque residual al resultado para reconstruir un bloque de imágenes. Los bloques reconstruidos resultantes

también pueden reenviarse a través del componente de filtros en bucle 425 al componente de memoria intermedia de imágenes decodificadas 423. El componente de memoria intermedia de imágenes decodificadas 423 continúa almacenando bloques de imágenes reconstruidos adicionales, que pueden reconstruirse en fotogramas por medio de la información de división. Dichos fotogramas también pueden ser colocados en una secuencia. La secuencia se envía hacia una pantalla como una señal de vídeo de salida reconstruida.

La figura 5 es un diagrama esquemático que ilustra un ejemplo de HRD 500. Se puede emplear un HRD 500 en un codificador, tal como el sistema de códec 200 y/o el codificador 300. El HRD 500 puede comprobar el flujo de bits creado en la etapa 109 del método 100 antes de que el flujo de bits se reenvíe a un decodificador, tal como el decodificador 400. En algunos ejemplos, el flujo de bits puede ser reenviado de manera continua a través del HRD 500 a medida que se codifica el flujo de bits. En el caso de que una parte del flujo de bits no cumpla con las restricciones asociadas, el HRD 500 puede indicar dicho fallo a un codificador para hacer que el codificador vuelva a codificar la sección correspondiente del flujo de bits con diferentes mecanismos.

El HRD 500 incluye un programador de flujo hipotético (Hypothetical Stream Scheduler, HSS) 541. Un HSS 541 es un componente configurado para realizar un mecanismo de entrega hipotético. El mecanismo de entrega hipotético se utiliza para comprobar la conformidad de un flujo de bits o un decodificador con respecto a la temporización y el flujo de datos de un flujo de bits 551 introducido en el HRD 500. Por ejemplo, el HSS 541 puede recibir una salida de flujo de bits 551 desde un codificador y gestionar el proceso de prueba de conformidad en el flujo de bits 551. En un ejemplo particular, el HSS 541 puede controlar la velocidad a la que las imágenes codificadas se mueven a través del HRD 500 y verificar que el flujo de bits 551 no contiene datos no conformes.

El HSS 541 puede reenviar el flujo de bits 551 a una CPB 543 a una velocidad predefinida. El HRD 500 puede gestionar datos en unidades de decodificación (Decoding Units, DU) 553. Una DU 553 es una Unidad de Acceso (AU) o un subconjunto de una AU y unidades de capa de abstracción de red (NAL) de capa no de codificación de vídeo (VCL) asociadas. Específicamente, una AU contiene una o más imágenes asociadas con un tiempo de salida. Por ejemplo, una AU puede contener una sola imagen en un flujo de bits de una sola capa y puede contener una imagen para cada capa en un flujo de bits de múltiples capas. Cada imagen de una AU se puede dividir en partes, cada una de las cuales se incluye en una unidad de NAL de VCL correspondiente. Por lo tanto, una DU 553 puede contener una o más imágenes, uno o más segmentos de una imagen o combinaciones de los mismas. Además, los parámetros utilizados para decodificar AU/DU, imágenes y/o segmentos se pueden incluir en unidades de NAL no de VCL. Por lo tanto, la DU 553 contiene unidades de NAL no de VCL que contienen datos necesarios para soportar la decodificación de las unidades de NAL de VCL en la DU 553. La CPB 543 es una memoria intermedia de primero en entrar, primero en salir, en el HRD 500. La CPB 543 contiene DU 553 que incluyen datos de vídeo en orden de decodificación. La CPB 543 almacena los datos de vídeo para su uso durante la verificación de conformidad del flujo de bits.

La CPB 543 reenvía las DU 553 a un componente de proceso de decodificación 545. El componente del proceso de decodificación 545 es un componente que se ajusta al estándar de VVC. Por ejemplo, el componente del proceso de decodificación 545 puede emular un decodificador 400 empleado por un usuario final. El componente del proceso de decodificación 545 decodifica las DU 553 a una velocidad que puede ser lograda mediante un decodificador de usuario final de ejemplo. Si el componente del proceso de decodificación 545 no puede decodificar las DU 553 lo suficientemente rápido como para evitar un desbordamiento (o evitar una insuficiencia de datos de la memoria intermedia) de la CPB 543, entonces el flujo de bits 551 no se ajusta al estándar y debe ser codificado de nuevo.

El componente del proceso de decodificación 545 decodifica las DU 553, lo que crea las DU decodificadas 555. Una DU 555 decodificada contiene una imagen decodificada. Las DU decodificadas 555 se reenvían a una DPB 547. La DPB 547 puede ser sustancialmente similar a un componente de memoria intermedia de imágenes decodificadas 223, 323 y/o 423. Para soportar la interpredicción, las imágenes que están marcadas para su uso como imágenes de referencia 556 que se obtienen de las DU decodificadas 555 se devuelven al componente del proceso de decodificación 545 para soportar una decodificación adicional. La DPB 547 genera la secuencia de vídeo decodificada como una serie de imágenes 557. Las imágenes 557 son imágenes reconstruidas que reflejan de manera general imágenes codificadas en el flujo de bits 551 por el codificador.

Las imágenes 557 se reenvían a un componente de recorte de salida 549. El componente de recorte de salida 549 está configurado para aplicar una ventana de recorte de conformidad a las imágenes 557. Esto da como resultado imágenes recortadas de salida 559. Una imagen recortada de salida 559 es una imagen completamente reconstruida. En consecuencia, la imagen recortada de salida 559 imita lo que vería un usuario final tras la decodificación del flujo de bits 551. Por lo tanto, el codificador puede revisar las imágenes recortadas de salida 559 para garantizar que la codificación sea satisfactoria.

El HRD 500 se inicializa basándose en los parámetros de HRD en el flujo de bits 551. Por ejemplo, el HRD 500 puede leer parámetros de HRD de un VPS, un SPS y/o mensajes de SEI. El HRD 500 puede entonces realizar operaciones de prueba de conformidad en el flujo de bits 551 basándose en la información de dichos

parámetros de HRD. Como ejemplo específico, el HRD 500 puede determinar uno o más cronogramas de entrega de CPB a partir de los parámetros de HRD. Un programa de entrega especifica el tiempo para la entrega de datos de vídeo hacia y/o desde una ubicación de memoria, tal como una CPB y/o un DPB. Por lo tanto, un cronograma de entrega de la CPB especifica el tiempo para la entrega de AU, DU 553 y/o imágenes hacia/desde la CPB 543. Cabe señalar que el HRD 500 puede emplear cronogramas de entrega de DPB para la DPB 547 que son similares a los cronogramas de entrega de CPB.

El vídeo puede codificarse en diferentes capas y/u OLS para su uso por decodificadores con diferentes niveles de capacidades de hardware, así como para diferentes condiciones de red. Los cronogramas de entrega de la CPB se seleccionan para reflejar estos problemas. En consecuencia, los subflujos de bits de capas superiores se designan para condiciones óptimas de hardware y red y, por lo tanto, las capas superiores pueden recibir uno o más programas de entrega de CPB que emplean una gran cantidad de memoria en la CPB 543 y retrasos cortos para las transferencias de las DU 553 hacia la DPB 547. Del mismo modo, los subflujos de bits de capa inferior están designados para capacidades limitadas del hardware del decodificador y/o malas condiciones de la red. Por lo tanto, las capas inferiores pueden recibir uno o más programas de entrega de CPB que emplean una pequeña cantidad de memoria en la CPB 543 y retrasos más largos para las transferencias de las DU 553 hacia la DPB 547. Los OLS, capas, subcapas o combinaciones de los mismos pueden entonces ser probados de acuerdo con el cronograma de entrega correspondiente para garantizar que el subflujo de bits resultante se pueda decodificar correctamente en las condiciones esperadas para el subflujo de bits. En consecuencia, los parámetros de HRD en el flujo de bits 551 pueden indicar los cronogramas de entrega de CPB así como incluir datos suficientes para permitir que el HRD 500 determine los cronogramas de entrega de CPB y correlacione los cronogramas de entrega de CPB con los OLS, capas y/o subcapas correspondientes.

La figura 6 es un diagrama esquemático que ilustra una secuencia de vídeo multicapa 600 de ejemplo, configurada para predicción entre capas 621. La secuencia de vídeo multicapa 600 puede ser codificada mediante un codificador, tal como el sistema de códec 200 y/o el codificador 300 y decodificada mediante un decodificador, tal como el sistema de códec 200 y/o el decodificador 400, por ejemplo, según el método 100. Además, un HRD, tal como el HRD 500, puede comprobar la conformidad estándar de la secuencia de vídeo multicapa 600. La secuencia de vídeo multicapa 600 se incluye para representar una aplicación de ejemplo para capas en una secuencia de vídeo codificada. Una secuencia de vídeo multicapa 600 es cualquier secuencia de vídeo que emplea una pluralidad de capas, tal como la capa N 631 y la capa N+1 632.

En un ejemplo, la secuencia de vídeo multicapa 600 puede emplear la predicción entre capas 621. La predicción entre capas 621 se aplica entre las imágenes 611, 612, 613 y 614 y las imágenes 615, 616, 617 y 618 en diferentes capas. En el ejemplo mostrado, las imágenes 611, 612, 613 y 614 son parte de la capa N+1 632 y las imágenes 615, 616, 617 y 618 son parte de la capa N 631. Una capa, tal como la capa N 631 y/o la capa N+1 632, es un grupo de imágenes que están todas asociadas con un valor similar de una característica, tal como un tamaño, calidad, resolución, relación señal/ruido similar, capacidad, etc. Una capa puede definirse formalmente como un conjunto de unidades de NAL de VCL y unidades asociadas de NAL no de VCL. Una unidad de NAL de VCL es una unidad de NAL codificada para contener datos de vídeo, tal como un segmento codificado de una imagen. Una unidad de NAL no de VCL es una unidad de NAL que contiene datos que no son de vídeo, tales como sintaxis y/o parámetros que soportan la decodificación de datos de vídeo, la realización de comprobaciones de conformidad u otras operaciones.

En el ejemplo mostrado, la capa N+1 632 está asociada con un tamaño de imagen mayor que la capa N 631. En consecuencia, las imágenes 611, 612, 613 y 614 en la capa N+1 632 tienen un tamaño de imagen mayor (por ejemplo, mayor altura y ancho y, por lo tanto, más muestras) que las imágenes 615, 616, 617 y 618 en la capa N 631 en este ejemplo. Sin embargo, dichas imágenes pueden estar separadas entre la capa N+1 632 y la capa N 631 por otras características. Si bien solo se muestran dos capas, la capa N+1 632 y la capa N 631, un conjunto de imágenes se puede separar en cualquier número de capas, según las características asociadas. La capa N+1 632 y la capa N 631 también pueden indicarse mediante un Id de capa. Un Id de capa es un elemento de datos asociado con una imagen e indica que la imagen es parte de una capa indicada. En consecuencia, cada imagen 611-618 puede asociarse con un Id de capa correspondiente para indicar qué capa N+1 632 o capa N 631 incluye la imagen correspondiente. Por ejemplo, un Id de capa puede incluir un identificador de capa de cabecera de unidad de NAL (nuh_layer_id), que es un elemento de sintaxis que especifica un identificador de una capa que incluye una unidad de NAL (por ejemplo, que incluye segmentos y/o parámetros de las imágenes en una capa). A una capa asociada con una calidad/tamaño de flujo de bits inferior, tal como la capa N 631, se le asigna de manera general un Id de capa inferior y se le denomina capa inferior. Además, a una capa asociada con una calidad/tamaño de flujo de bits superior, tal como la capa N+1 632, se le asigna de manera general un Id de capa superior y se la denomina capa superior.

Las imágenes 611-618 en diferentes capas 631-632 están configuradas para ser mostradas como alternativa. Como ejemplo específico, un decodificador puede decodificar y mostrar la imagen 615 en el momento de visualización actual si se desea una imagen más pequeña, o el decodificador puede decodificar y mostrar la imagen 611 en el momento de visualización actual si se desea una imagen más grande. Por lo tanto, las imágenes 611-614 en la capa superior N+1 632 contienen sustancialmente los mismos datos de imagen que

las imágenes correspondientes 615-618 en la capa inferior N 631 (a pesar de la diferencia en el tamaño de la imagen). Específicamente, la imagen 611 contiene sustancialmente los mismos datos de imagen que la imagen 615, la imagen 612 contiene sustancialmente los mismos datos de imagen que la imagen 616, etc.

Las imágenes 611-618 se pueden codificar con referencia a otras imágenes 611-618 en la misma capa N 631 o N+1 632. Codificar una imagen en referencia a otra imagen en la misma capa da como resultado la interpredicción 623. La interpredicción 623 se representa con flechas en línea continua. Por ejemplo, la imagen 613 puede codificarse empleando la interpredicción 623 usando una o dos de las imágenes 611, 612 y/o 614 en la capa N+1 632 como referencia, donde se hace referencia a una imagen para la interpredicción unidireccional y/o se hace referencia a dos imágenes para la interpredicción bidireccional. Por ejemplo, la imagen 617 puede codificarse empleando la interpredicción 623 usando una o dos de las imágenes 615, 616 y/o 618 en la capa N 631 como referencia, donde se hace referencia a una imagen para la interpredicción unidireccional y/o se hace referencia a dos imágenes para la interpredicción bidireccional. Cuando se utiliza una imagen como referencia para otra imagen en la misma capa al realizar la interpredicción 623, la imagen puede denominarse imagen de referencia. Por ejemplo, la imagen 612 puede ser una imagen de referencia utilizada para la codificación de la imagen 613 según la interpredicción 623. La interpredicción 623 también puede denominarse predicción de intra-capa en un contexto multicapa. Por lo tanto, la interpredicción 623 es un mecanismo de codificación de muestras de una imagen actual por referencia a muestras indicadas en una imagen de referencia que es diferente de la imagen actual, donde la imagen de referencia y la imagen actual están en la misma capa.

Las imágenes 611-618 también se pueden codificar con referencia a otras imágenes 611-618 en diferentes capas. Este proceso se conoce como predicción entre capas 621 y se representa mediante flechas discontinuas. La predicción entre capas 621 es un mecanismo de codificación de muestras de una imagen actual por referencia a muestras indicadas en una imagen de referencia, donde la imagen actual y la imagen de referencia están en diferentes capas y, por lo tanto, tienen diferentes ID de capa. Por ejemplo, una imagen en una capa inferior N 631 se puede utilizar como imagen de referencia de codificación de una imagen correspondiente en una capa superior N+1 632. Como ejemplo específico, la imagen 611 se puede codificar con referencia a la imagen 615, según la predicción entre capas 621. En tal caso, la imagen 615 se utiliza como imagen de referencia entre capas. Una imagen de referencia entre capas es una imagen de referencia utilizada para la predicción entre capas 621. En la mayoría de los casos, la predicción entre capas 621 está restringida de tal manera que una imagen actual, tal como la imagen 611, solo puede usar una o varias imágenes de referencia entre capas que están incluidas en la misma AU y que están en una capa inferior, tal como la imagen 615. Cuando están disponibles múltiples capas (por ejemplo, más de dos), la predicción entre capas 621 puede codificar/decodificar una imagen actual basándose en múltiples imágenes de referencia entre capas en niveles más bajos que la imagen actual.

Un codificador de vídeo puede emplear una secuencia de vídeo multicapa 600 de codificación de imágenes 611-618 a través de muchas combinaciones y/o permutaciones diferentes de interpredicción 623 y predicción entre capas 621. Por ejemplo, la imagen 615 puede codificarse según una intrapredicción. Las imágenes 616-618 pueden codificarse entonces según la interpredicción 623 utilizando la imagen 615 como imagen de referencia. Además, la imagen 611 puede codificarse según la predicción entre capas 621 utilizando la imagen 615 como imagen de referencia entre capas. Las imágenes 612-614 pueden codificarse entonces según la interpredicción 623 utilizando la imagen 611 como imagen de referencia. Por lo tanto, una imagen de referencia puede servir como imagen de referencia de una sola capa y como imagen de referencia entre capas para diferentes mecanismos de codificación. Codificando imágenes de capa superior N+1 632 basadas en imágenes de capa inferior N 631, la capa superior N+1 632 puede evitar el empleo de intrapredicción, que tiene una eficiencia de codificación mucho menor que la interpredicción 623 y la predicción entre capas 621. Por lo tanto, la baja eficiencia de codificación de la intrapredicción puede limitarse a las imágenes más pequeñas/de menor calidad y, por lo tanto, limitarse a codificar la menor cantidad de datos de vídeo. Las imágenes utilizadas como imágenes de referencia y/o imágenes de referencia entre capas pueden indicarse en entradas de una o varias listas de imágenes de referencia contenidas en una estructura de lista de imágenes de referencia.

Cabe señalar que las capas, tales como la capa N+1 632 y la capa N 631, pueden incluirse en conjuntos de capas de salida (Output Layer Sets, OLS). Un OLS es un conjunto de una o más capas, donde al menos una capa es una capa de salida. Por ejemplo, la capa N 631 puede incluirse en un primer OLS y la capa N 631 y la capa N-1 632 pueden incluirse ambas en un segundo OLS. Esto permite enviar diferentes OLS a diferentes decodificadores, dependiendo de las condiciones del lado del decodificador. Por ejemplo, un proceso de extracción de subflujo de bits puede eliminar datos que no están relacionados con un OLS objetivo de la secuencia de vídeo multicapa 600 antes de que el OLS objetivo se envíe a un decodificador. Por lo tanto, se puede almacenar una copia codificada de la secuencia de vídeo multicapa 600 en un codificador (o en un servidor de contenidos correspondiente), y se pueden extraer varios OLS y enviarlos a diferentes decodificadores previa solicitud.

Una capa de transmisión simultánea es una capa que no emplea la predicción entre capas 621. Por ejemplo, la capa N+1 632 se codifica con referencia a la capa N 631 basándose en la predicción entre capas 621. Sin

embargo, la capa 631 no está codificada con referencia a otra capa. Por lo tanto, la capa 631 es una capa de transmisión simultánea. Las secuencias de vídeo escalables, tales como la secuencia de vídeo multicapa 600, emplean de manera general una capa base y una o más capas de mejora que mejoran alguna propiedad de la capa base. En la figura 6, la capa N 631 es una capa base. Una capa base se codifica de manera general como una capa de transmisión simultánea. También cabe señalar que la figura 6 es ejemplar y no limitante ya que las secuencias de vídeo con múltiples capas pueden usar muchas combinaciones/permutaciones diferentes de dependencias. Un flujo de bits puede contener cualquier número de capas, y cualquier número de dichas capas puede ser capas de transmisión simultánea. Por ejemplo, la predicción entre capas 621 se puede omitir por completo, en cuyo caso todas las capas son capas de transmisión simultánea. Como ejemplo adicional, las aplicaciones multivista muestran dos o más capas de salida. Por lo tanto, una aplicación de múltiples vistas incluye de manera general dos o más capas base, que son capas de transmisión simultánea, y pueden incluir capas de mejora correspondientes a cada capa base.

Las capas de transmisión simultánea pueden manejarse de manera diferente a las capas que usan la predicción entre capas 621. Por ejemplo, cuando se codifican capas que usan predicción entre capas 621, un codificador debería indicar el número de capas así como las dependencias entre las capas, con el fin de soportar la decodificación. Sin embargo, dicha información se puede omitir para las capas de transmisión simultánea. Por ejemplo, la configuración de la capa N+1 632 y la capa N 631 se puede indicar en un VPS, tal como se explica con más detalle a continuación. Sin embargo, la capa N 631 puede decodificarse sin dicha información. Por lo tanto, el VPS puede eliminarse de un flujo de bits correspondiente cuando solo se transmite la capa N 631 a un decodificador. Sin embargo, esto puede crear errores si los parámetros restantes en el flujo de bits hacen referencia al VPS. Estas y otras cuestiones se explican con mayor detalle a continuación.

La figura 7 es un diagrama esquemático que ilustra un flujo de bits 700 de ejemplo. Por ejemplo, el flujo de bits 700 puede generarse mediante un sistema de códec 200 y/o un codificador 300, para decodificarlo mediante un sistema de códec 200 y/o un decodificador 400, según el método 100. Además, el flujo de bits 700 puede incluir una secuencia de vídeo multicapa 600. Además, el flujo de bits 700 puede incluir diversos parámetros para controlar el funcionamiento de un HRD, tal como el HRD 500. Basándose en dichos parámetros, el HRD 500 puede comprobar que el flujo de bits 700 cumple con los estándares antes de la transmisión hacia un decodificador para su decodificación.

El flujo de bits 700 incluye un VPS 711, uno o más SPS 713, una pluralidad de conjuntos de parámetros de imagen (Picture Parameter Set, PPS) 715, una pluralidad de cabeceras de segmento 717 y datos de imagen 720. Un VPS 711 contiene datos relacionados con todo el flujo de bits 700. Por ejemplo, el VPS 711 puede contener varios OLS, capas y/o subcapas relacionados con datos utilizados en el flujo de bits 700. Un SPS 713 contiene datos de secuencia comunes a todas las imágenes en una secuencia de vídeo codificada contenida en el flujo de bits 700. Por ejemplo, cada capa puede contener una o más secuencias de vídeo codificadas, y cada secuencia de vídeo codificada puede hacer referencia a un SPS 713 para los parámetros correspondientes. Los parámetros en un SPS 713 pueden incluir tamaño de imagen, profundidad de bits, parámetros de herramientas de codificación, restricciones de velocidad de bits, etc. Cabe señalar que, si bien cada secuencia se refiere a un SPS 713, un solo SPS 713 puede contener datos para múltiples secuencias en algunos ejemplos. El PPS 715 contiene parámetros que se aplican a una imagen completa. Por lo tanto, cada imagen de la secuencia de vídeo puede hacer referencia a un PPS 715. Cabe señalar que, si bien cada imagen hace referencia a un PPS 715, un solo PPS 715 puede contener datos para múltiples imágenes en algunos ejemplos. Por ejemplo, se pueden codificar múltiples imágenes similares según parámetros similares. En tal caso, un solo PPS 715 puede contener datos para imágenes similares. El PPS 715 puede indicar herramientas de codificación disponibles para segmentos en las imágenes correspondientes, parámetros de cuantificación, compensaciones, etc.

La cabecera de segmento 717 contiene parámetros que son específicos de cada segmento en una imagen. Por lo tanto, puede haber una cabecera de segmento 717 por cada segmento en la secuencia de vídeo. La cabecera de segmento 717 puede contener información de tipo de segmento, recuentos de orden de imágenes (POC), listas de imágenes de referencia, pesos de predicción, puntos de entrada de mosaicos, parámetros de desbloqueo, etc. Cabe señalar que en algunos ejemplos, un flujo de bits 700 también puede incluir una cabecera de imagen, que es una estructura de sintaxis que contiene parámetros que se aplican a todos los segmentos de una sola imagen. Por esta razón, una cabecera de imagen y una cabecera de segmento 717 pueden usarse indistintamente en algunos contextos. Por ejemplo, ciertos parámetros pueden ser movidos entre la cabecera de segmento 717 y una cabecera de imagen dependiendo de si dichos parámetros son comunes a todos los segmentos en una imagen.

Los datos de imagen 720 contienen datos de vídeo codificados según interpredicción, predicción entre capas y/o intrapredicción, así como datos residuales transformados y cuantificados correspondientes. Por ejemplo, los datos de imagen 720 pueden incluir capas 723 y 724, imágenes 725 y 726 y/o segmentos 727 y 728. Una capa 723 y 724 es un conjunto de unidades de NAL de VCL 741 que comparten una característica específica (por ejemplo, una resolución, velocidad de fotogramas, tamaño de imagen, etc. comunes) tal como se indica mediante un ID de capa, tal como un nuh_layer_id 732, y unidades de NAL no de VCL 742 asociadas. Por

ejemplo, una capa 723 puede incluir un conjunto de imágenes 725 que comparten el mismo nuh_layer_id 732. Asimismo, una capa 724 puede incluir un conjunto de imágenes 726 que comparten el mismo nuh_layer_id 732. Las capas 723 y 724 pueden ser sustancialmente similares, pero pueden contener contenido diferente. Por ejemplo, las capas 723 y 724 pueden contener la capa N 631 y la capa N+1 632, respectivamente, de la figura 6. Por lo tanto, un flujo de bits codificado 700 puede incluir múltiples capas 723 y 724. Si bien solo se muestran dos capas 723 y 724 para mayor claridad de la explicación, se puede incluir cualquier número de capas 723 y 724 en el flujo de bits 700.

Un nuh_layer_id 732 es un elemento de sintaxis que especifica un identificador de una capa 723 y/o 724 que incluye al menos una unidad de NAL. Por ejemplo, una capa de menor calidad, conocida como capa base, puede incluir el valor más bajo de nuh_layer_id 732 con valores crecientes de nuh_layer_id 732 para capas de mayor calidad. Por lo tanto, una capa inferior es una capa 723 o 724 con un valor menor de nuh_layer_id 732 y una capa superior es una capa 723 o 724 con un valor mayor de nuh_layer_id 732. Los datos de las capas 723 y 724 se correlacionan basándose en el nuh_layer_id 732. Por ejemplo, los conjuntos de parámetros y datos de vídeo pueden asociarse con un valor de nuh_layer_id 732 que corresponde a la capa más baja 723 o 724 que incluye dichos conjuntos de parámetros/datos de vídeo. Por lo tanto, un conjunto de unidades de NAL de VCL 741 son parte de una capa 723 y/o 724 cuando el conjunto de unidades de NAL de VCL 741 tienen todas un valor particular de nuh_layer_id 732.

Una imagen 725 y 726 es una matriz de muestras de luma y/o una matriz de muestras de croma que crean un fotograma o un campo del mismo. Por ejemplo, una imagen 725/726 es una imagen codificada que puede ser generada para su visualización o ser utilizada para soportar la codificación de otra u otras imágenes para su emisión. Las imágenes 725 y 726 son sustancialmente similares, pero la imagen 725 está contenida en la capa 723 mientras que la imagen 726 está contenida en la capa 724. Una imagen 725 y 726 contiene uno o más segmentos 727 y 728, respectivamente. Un segmento 727/728 puede definirse como un número entero de mosaicos completos o un número entero de filas consecutivas de unidades de árbol de codificación (CTU) completas (por ejemplo, dentro de un mosaico) de una imagen 725/726 que están contenidas exclusivamente en una sola unidad de NAL, como una unidad de NAL de VCL 741. Los segmentos 727 y los segmentos 728 son sustancialmente similares, excepto por que los segmentos 727 están incluidos en las imágenes 725 y la capa 723, mientras que los segmentos 728 están incluidos en las imágenes 726 y la capa 724. Los segmentos 727/728 se dividen además en CTU y/o bloques de árbol de codificación (CTB). Una CTU es un grupo de muestras de un tamaño predefinido que puede dividirse mediante un árbol de codificación. Un CTB es un subconjunto de una CTU y contiene componentes de luma o componentes de croma de la CTU. Las CTU/CTB se dividen además en bloques de codificación basados en árboles de codificación. Los bloques de codificación pueden luego ser codificados/decodificados según mecanismos de predicción.

Un flujo de bits 700 puede codificarse como una secuencia de unidades de NAL. Una unidad de NAL es un bin de datos de vídeo y/o sintaxis de soporte. Una unidad de NAL puede ser una unidad de NAL de VCL 741 o una unidad de NAL no de VCL 742. Una unidad de NAL de VCL 741 es una unidad de NAL codificada para contener datos de vídeo, tales como datos de imagen 720 y una cabecera de segmento 717 asociada. Como ejemplo específico, cada segmento 727 y 728 y una cabecera de segmento 717 asociada se pueden codificar en una sola unidad de NAL de VCL 741. Una unidad de NAL no de VCL 742 es una unidad de NAL que contiene datos que no son de vídeo, tales como sintaxis y/o parámetros que soportan la decodificación de datos de vídeo, la realización de comprobaciones de conformidad u otras operaciones. Por ejemplo, una unidad de NAL 742 no de VCL puede contener un VPS 711, un SPS 713, un PPS 715, una cabecera de imagen u otra sintaxis de soporte. Por lo tanto, un flujo de bits 700 es una serie de unidades de NAL de VCL 741 y unidades de NAL no de VCL 742. Cada unidad de NAL contiene un nuh_layer_id 732, que permite a un codificador o decodificador determinar qué capa 723 o 724 incluye la unidad de NAL correspondiente.

Un flujo de bits 700 que incluye múltiples capas 723 y 724 puede codificarse y almacenarse hasta que lo solicite un decodificador. Por ejemplo, un decodificador puede solicitar una capa 723, una capa 724 y/o un OLS que contenga múltiples capas 723 y 724. En un ejemplo particular, la capa 723 es una capa base y la capa 724 es una capa de mejora. También se pueden emplear capas adicionales en el flujo de bits 700. El codificador y/o un servidor de contenido deben enviar solo las capas 723 y/o 724 al decodificador, que son necesarias para decodificar la una o varias capas de salida solicitadas. Por ejemplo, cuando se usan capas para diferentes tamaños de imagen, un decodificador que solicite el tamaño de imagen más grande puede recibir el flujo de bits 700 completo con ambas capas 723 y 724. Un decodificador que solicite el tamaño de imagen más pequeño puede recibir las únicas capas 723. Un decodificador que solicita un tamaño de imagen intermedio puede recibir la capa 723 y otras capas intermedias pero no la capa más alta 724 y, por lo tanto, no todo el flujo de bits. También se puede utilizar el mismo enfoque para otras características de la capa, tales como la velocidad de fotogramas, la resolución de la imagen, etc.

Se emplea un proceso de extracción de subflujo de bits 729 para extraer un subflujo de bits 701 del flujo de bits 700 para soportar la funcionalidad descrita anteriormente. Un subflujo de bits 701 es un subconjunto de las unidades de NAL (por ejemplo, unidades de NAL no de VCL 742 y unidades de NAL de VCL 741) del flujo de bits 700. Específicamente, un subflujo de bits 701 puede contener datos relacionados con una o más capas,

pero no datos relacionados con otras capas. En el ejemplo mostrado, el subflujo de bits 701 contiene datos relacionados con la capa 723, pero no datos relacionados con la capa 724. Por lo tanto, el subflujo de bits 701 contiene varios SPS 713, PPS 715, cabeceras de segmento 717 y datos de imagen 720 que incluyen capa 723, imágenes 725 y segmentos 727. El proceso de extracción de subflujo de bits 729 elimina unidades NAL basadas en nuh_layer_id 732. Por ejemplo, las unidades de NAL de VCL 741 y las unidades de NAL no de VCL 742 asociadas solo con la capa superior 724 incluyen valores de nuh_layer_id 732 más altos y, por lo tanto, eliminando todas las unidades de NAL con los valores de nuh_layer_id 732 más altos se extrae la capa inferior 723 y los parámetros asociados. Cada unidad de NAL contiene un valor de nuh_layer_id 732 que es menor o igual que el nuh_layer_id 732 de la capa más baja que incluye la unidad de NAL para soportar el proceso de extracción de subflujo de bits 729. Cabe señalar que un flujo de bits 700 y un subflujo de bits 701 pueden denominarse cada uno, de manera general, flujo de bits.

En el ejemplo mostrado, el subflujo de bits 701 incluye una capa de transmisión simultánea (por ejemplo, una capa base). Tal como se señaló anteriormente, una capa de transmisión simultánea es cualquier capa que no utiliza predicción entre capas. El VPS 711 contiene datos que describen la configuración de las capas 723 y 724. Sin embargo, estos datos no son necesarios para decodificar una capa de transmisión simultánea, tal como la capa 723. Por lo tanto, el proceso de extracción de subflujo de bits 729 elimina el VPS 711 para soportar una mayor eficiencia de codificación cuando se extrae una capa de transmisión simultánea. Esto puede causar problemas en algunos sistemas de codificación de vídeo. Específicamente, ciertos parámetros en el SPS 713 pueden hacer referencia al VPS 711. Cuando se elimina el VPS 711, es posible que el decodificador y/o el HRD no sean capaces de resolver dichos parámetros, ya que los datos a los que hacen referencia dichos parámetros ya no están presentes. Esto puede provocar un error al realizar pruebas de conformidad en el HRD en capas de transmisión simultánea. Alternativamente, esto puede dar como resultado errores impredecibles en un decodificador cuando se transmite una capa de transmisión simultánea para su visualización en el decodificador.

La presente divulgación aborda estos errores. Específicamente, el SPS 713 incluye un sps_video_parameter_set_id 731. El sps_video_parameter_set_id 731 es un elemento de sintaxis que especifica un ID de una referencia del VPS 711 por parte del SPS 713. Específicamente, el VPS 711 contiene un vps_video_parameter_set_id 735, que es un elemento de sintaxis que proporciona un ID para el VPS 711 como referencia para otros elementos/estructuras de sintaxis. Cuando el VPS 711 está presente, el sps_video_parameter_set_id 731 se ajusta al valor de vps_video_parameter_set_id 735. Sin embargo, cuando se usa el SPS 713 para una capa de transmisión simultánea, el sps_video_parameter_set_id 731 se ajusta a cero. Dicho de otra manera, el sps_video_parameter_set_id 731, cuando es mayor que cero, especifica el valor de vps_video_parameter_set_id 735 para el VPS 711 al que hace referencia el SPS 713. Cuando el sps_video_parameter_set_id 731 es igual a cero, el SPS 713 no hace referencia a un VPS 711, y no se hace referencia a ningún VPS 711 cuando se decodifica cualquier secuencia de vídeo de capa codificada que haga referencia al SPS 713. Esto se puede lograr usando un SPS 713 separado para diferentes capas (por ejemplo, un SPS para una capa de transmisión simultánea y otro SPS para una capa o capas que no son de transmisión simultánea) o cambiando el valor de sps_video_parameter_set_id 731 durante el proceso de extracción de subflujo de bits 729. De esta manera, el sps_video_parameter_set_id 731 no hace referencia erróneamente a un ID que no esté disponible cuando se elimina el VPS 711 durante el proceso de extracción de subflujo de bits 729.

Además, diversas variables que son obtenidas por el HRD y/o por el decodificador también hacen referencia a parámetros en el VPS 711. En consecuencia, dichas variables se ajustan a valores predeterminados cuando el sps_video_parameter_set_id 731 se ajusta a cero. Esto garantiza que dichas variables puedan resolverse adecuadamente a un valor procesable cuando se extrae el VPS 711 para capas de transmisión simultánea, mientras sigue funcionando correctamente para flujos de bits multicapa. Por ejemplo, un decodificador y/o un HRD pueden obtener un GeneralLayerIdx[i] basándose en un flujo de bits 700 y/o en un subflujo de bits 701. Un índice de capa general (GeneralLayerIdx[i]) es una variable derivada que especifica un índice de una capa i correspondiente. Por lo tanto, el GeneralLayerIdx[i] se puede emplear para determinar un índice de capa de una capa actual incluyendo el nuh_layer_id 732 de la capa actual como capa i en GeneralLayerIdx[i]. Esto se puede expresar como un índice de capa general correspondiente a un id de capa de nuh (GeneralLayerIdx[nuh_layer_id]). Por lo tanto, el GeneralLayerIdx[nuh_layer_id] indica un índice de capa actual para una capa correspondiente. Este proceso funciona correctamente para capas que no son de transmisión simultánea, tal como la capa 724, pero puede causar errores en la capa de transmisión simultánea 723. En consecuencia, el GeneralLayerIdx[nuh_layer_id] se ajusta a, y/o se infiere que es, cero cuando el sps_video_parameter_set_id 731 es cero (lo que indica una capa de transmisión simultánea).

Como ejemplo adicional, el VPS 711 puede contener un indicador de capa independiente de VPS (vps_independent_layer_flag) 733. El vps_independent_layer_flag 733 especifica si las capas correspondientes, tales como la capa 723 y/o 724, usan predicción entre capas. En consecuencia, vps_independent_layer_flag [GeneralLayerIdx[nuh_layer_id]] especifica si una capa actual con índice GeneralLayerIdx[nuh_layer_id] utiliza predicción entre capas. Sin embargo, el VPS 711 que contiene el vps_independent_layer_flag 733 no se envía al decodificador cuando la capa 723 enviada al decodificador es

una capa de transmisión simultánea. Por lo tanto, la referencia puede provocar un error. Sin embargo, las capas de transmisión simultánea no utilizan predicción entre capas. Por lo tanto, se puede inferir que el vps_independent_layer_flag 733 para una capa de transmisión simultánea es igual a uno, lo que indica que no se usa ninguna predicción entre capas para una capa correspondiente 723. Por lo tanto, vps_independent_layer_flag[GeneralLayerIdx[nuh_layer_id]] se ajusta a/se infiere que es uno, para indicar que no hay predicción entre capas cuando el sps_video_parameter_set_id está ajustado a cero. De esta manera, se evitan errores cuando se elimina un VPS de un flujo de bits antes de la transmisión de una capa de transmisión simultánea, tal como la capa 723. Como resultado, se incrementa la funcionalidad del codificador y del decodificador. Además, la eficiencia de la codificación aumenta al eliminar con éxito un VPS innecesario de un flujo de bits que incluye solo una capa de transmisión simultánea, lo que reduce el uso de recursos de procesador, memoria y/o señalización de red, tanto en el codificador como en el decodificador.

La información anterior se describe a continuación con más detalle en lo que sigue en la presente memoria. La codificación de vídeo en capas también se conoce como codificación de vídeo escalable o codificación de vídeo con escalabilidad. La escalabilidad en la codificación de vídeo se puede favorecer mediante el uso de técnicas de codificación multicapa. Un flujo de bits multicapa comprende una capa base (Base Layer, BL) y una o más capas de mejora (Enhancement Layer, EL). Ejemplos de escalabilidad incluyen escalabilidad espacial, escalabilidad de calidad/relación señal a ruido (Signal to Noise Ratio, SNR), escalabilidad de múltiples vistas, escalabilidad de velocidad de fotogramas, etc. Cuando se utiliza una técnica de codificación multicapa, una imagen o una parte de la misma se puede codificar sin utilizar una imagen de referencia (intrapredicción), se puede codificar haciendo referencia a imágenes de referencia que están en la misma capa (interpredicción) y/o se puede codificar haciendo referencia a imágenes de referencia que están en otra u otras capas (predicción entre capas). Una imagen de referencia usada para la predicción entre capas de la imagen actual se denomina imagen de referencia entre capas (Inter-Layer Reference Picture, ILRP). La figura 6 ilustra un ejemplo de codificación multicapa para escalabilidad espacial en el que imágenes en diferentes capas tienen diferentes resoluciones.

Algunas familias de codificación de vídeo brindan soporte para escalabilidad en uno o varios perfiles separados de los uno o varios perfiles para la codificación de una sola capa. La codificación de vídeo escalable (SVC) es una extensión escalable de la codificación de vídeo avanzada (AVC) que brinda soporte para escalabilidades espaciales, temporales y de calidad. Para SVC, se señaliza un indicador en cada macrobloque (MB) en las imágenes de EL para indicar si el MB de EL se predice usando el bloque coubicado de una capa inferior. La predicción del bloque coubicado puede incluir textura, vectores de movimiento y/o modos de codificación. Las implementaciones de SVC no pueden reutilizar directamente implementaciones de AVC no modificadas en su diseño. La sintaxis y el proceso de decodificación de macrobloque de EL de SVC difieren de la sintaxis y el proceso de decodificación de AVC.

HEVC escalable (SHVC) es una extensión de HEVC que proporciona soporte para escalabilidades espaciales y de calidad. HEVC multivista (MV-HEVC) es una extensión de HEVC que brinda soporte para la escalabilidad de múltiples vistas. HEVC 3D (3D-HEVC) es una extensión de HEVC que brinda soporte para codificación de vídeo 3D, que es más avanzada y más eficiente que MV-HEVC. La escalabilidad temporal puede incluirse como parte integral de un códec de HEVC de una sola capa. En la extensión multicapa de HEVC, las imágenes decodificadas utilizadas para la predicción entre capas provienen solamente de la misma AU y se tratan como imágenes de referencia a largo plazo (Long-Term Reference Picture, LTRP). A dichas imágenes se les asignan índices de referencia en la una o varias listas de imágenes de referencia junto con otras imágenes de referencia temporales en la capa actual. La predicción entre capas (Inter-layer Prediction, ILP) se logra al nivel de unidad de predicción (PU) estableciendo el valor del índice de referencia para referirse a la una o varias imágenes de referencia entre capas en la una o varias listas de imágenes de referencia. La escalabilidad espacial vuelve a muestrear una imagen de referencia o parte de ella cuando una ILRP tiene una resolución espacial diferente a la imagen actual que se está codificando o decodificando. El remuestreo de la imagen de referencia se puede realizar a nivel de imagen o a nivel de bloque de codificación.

VVC también puede soportar codificación de vídeo en capas. Un flujo de bits de VVC puede incluir varias capas. Las capas pueden ser todas independientes entre sí. Por ejemplo, cada capa se puede codificar sin utilizar predicción entre capas. En este caso, las capas también se denominan capas de transmisión simultánea. En algunos casos, algunas de las capas se codifican mediante ILP. Una indicador en el VPS puede indicar si las capas son capas de transmisión simultánea o si algunas capas usan ILP. Cuando algunas capas utilizan ILP, la relación de dependencia entre capas también se señaliza en el VPS. A diferencia de SHVC y MV-HEVC, es posible que VVC no especifique los OLS. Un OLS incluye un conjunto específico de capas, donde una o más capas del conjunto de capas se especifican como capas de salida. Una capa de salida es una capa de un OLS que es generada. En algunas implementaciones de VVC, solo se puede seleccionar una capa para decodificación y emisión cuando las capas son capas de transmisión simultánea. En algunas implementaciones de VVC, se especifica que todo el flujo de bits, incluidas todas las capas, se decodificará cuando cualquier capa utilice ILP. Además, se especifica que ciertas capas entre las capas sean capas de salida. Se puede indicar que las capas de salida son solo la capa más alta, todas las capas o la capa más alta más un conjunto de capas inferiores indicadas.

Los aspectos anteriores contienen ciertos problemas relacionados con la escalabilidad. El diseño de escalabilidad en tales sistemas incluye perfil, escalonado y nivel (Profile, Tier, and Level, PTL) específicos de cada capa, así como operaciones de memoria intermedia de imágenes codificadas (Coded Picture Buffer, CPB) específicas de cada capa. Se debe mejorar la eficiencia de la señalización de PTL. Se debe mejorar la eficiencia de la señalización de los parámetros de HRD a nivel de secuencia para subcapas. Se debe mejorar la señalización de los parámetros de DPB. Algunos diseños hacen que los flujos de bits de una sola capa se refieran a los VPS. El rango de valores de num_ref_entries[ ][ ] en dichos diseños es incorrecto y provoca errores inesperados en los decodificadores. El proceso de decodificación en dichos diseños implica la extracción de subflujos de bits, lo que añade una carga a las implementaciones de decodificador. Es posible que el proceso de decodificación general para tales diseños no funcione para flujos de bits escalables que contienen múltiples capas con predicción entre capas. La obtención del valor de la variable NoOutputOfPriorPicsFlag en dichos diseños puede basarse en imágenes y no en AU. El mensaje de SEI de anidamiento escalable en tales diseños debe simplificarse para aplicarse directamente a los OLS, en lugar de a capas de los OLS, cuando nesting_ols_flag es igual a uno. Un mensaje de SEI anidado no escalable, cuando payloadType (tipo de carga útil) es igual a cero (período de almacenamiento en memoria intermedia), uno (temporización de la imagen) o ciento treinta (información de la unidad de decodificación), se puede especificar para que se aplique solo al OLS de orden 0.

En general, esta divulgación describe diversos enfoques para la escalabilidad en la codificación de vídeo. Las descripciones de las técnicas se basan en VVC. Sin embargo, las técnicas también se aplican a la codificación de vídeo en capas basándose en otras especificaciones de códec de vídeo. Uno o más de los problemas mencionados anteriormente se pueden resolver de la siguiente manera. Específicamente, esta divulgación incluye métodos para mejorar el soporte de escalabilidad en la codificación de vídeo.

Las siguientes son diversas definiciones de ejemplo. Un OP puede ser un subconjunto temporal de un OLS, identificado por un índice de OLS y un valor más alto de TemporalId (Id temporal). Una capa de salida es una capa de un OLS que es generada. Un OLS puede ser un conjunto de capas, donde una o más capas del conjunto de capas se especifican como capas de salida. Un índice de capa de OLS puede ser un índice de una capa en un OLS a la lista de capas en el OLS. Un proceso de extracción de subflujo de bits puede ser un proceso específico mediante el cual las unidades de NAL en un flujo de bits que no pertenecen a un conjunto de destino, determinado por un índice de OLS objetivo y un TemporalId objetivo más alto, se eliminan del flujo de bits, comprendiendo el subflujo de bits de salida las unidades NAL en el flujo de bits que pertenece al conjunto de destino.

Un ejemplo de sintaxis de RBSP de conjunto de parámetros de vídeo es el siguiente.

Video_parameter_set_rbsp( ) {	Descriptor
vps_video_parameter_set_id	u(4)
vps_max_layers_minus1	u(6)
vps_max_sub_layers_minus1	u(3)
si (vps_max_layers_minus1 > 0 && vps_max_sub_layers_minus1 > 0)	
vps_all_layers_same_num_sub_layers_flag	u(1)
si (vps_max_layers_minus1 > 0)	
vps_all_independent_layers_flag	u(1)
...	
vps_num_ptls	u(8)
para (i = 0; i < vps_num_ptls; i++) {	
si (i>0)	
pt_present_flag[i]	u(1)
si (vps_max_sub_layers_minus1 > 0 *&&* !vps_all_layers_same_num_sub_layers_flag)	
ptl_max_temporal_id[i]	u(3)
}	
mientras (!byte_aligned( ))	
vps_ptl_byte_alignment_zero_bit /* igual a 0 */	u(1)
para (i = 0; i < vps_num_ptls; i++)	
profile_tier_level(pt_present_flag[i], ptl_max_temporal_id[i])	

para(i = 0; i < TotalNumOlss; i++)	
si (NumLayersInOls[i] > 1 && vps_num_ptls > 1)	
ols_ptl_idx[i]	u(8)
si (!vps_all_independent_layers_flag)	
vps_num_dpb_params	ue(v)
si (vps_num_dpb_params > 0) {	
same_dpb_size_output_or_nonoutput_flag	u(1)
si (vps_max_sub_layers_minus1 > 0)	
vps_sub_layer_dpb_params_present_flag	u(1)
}	
para (i = 0; i <vps_num_dpb_params; i++) {	
dpb_size_only_flag[i]	u(1)
si (vps_max_sub_layers_minus1 > 0 && !vps_all_layers_same_num_sub_layers_flag)	
dpb_max_temporal_id[i]	u(3)
dpb_parameters(dpb_size_only_flag[i], dpb_max_temporal_id[i], vps_sub_layer_dpb_params_present_flag)	
}	
para (i = 0; i < vps_max_layers minus1 && vps_num_dpb_params > 1; i++) {	
si (!vps_independent_layer_flag[i])	
layer_output_dpb_params_idx[i]	ue(v)
si (LayerUsedAsRefLayerFlag[i] && !same_dpb_size_output_or_nonoutput_flag)	
layer_nonoutput_dpb_params_idx[i]	ue(v)
}	
general_hrd_params_present_flag	u(1)
si (general_hrd_params_present_flag) {	
num_units_in_tick	u(32)
time_scale	u(32)
general_hrd_parameters( )	
}	
vps_extension_flag	u(1)
si (vps_extension_flag)	
mientras (more_rbsp_data ( ))	
vps_extension_data_flag	u(1)
rbsp_trailing_bits( )	
}	

Un ejemplo de sintaxis de RBSP de conjunto de parámetros de secuencia es el siguiente.

seq_parameter_set_rbsp( ) {	Descriptor
sps_decoding_parameter_set_id	u(4)
sps_video_parameter_set_id	u(4)
sps_max_sub_layers_minus1	u(3)
sps_reserved_zero_4bits	u(4)
sps_ptl_dpb_present_flag	u(1)
si (sps_ptl_dpb_present_flag)	
profile_tier_level(1, sps_max_sub_layers_minus1)	
gdr_enabled_flag	u(1)
sps_seq_parameter_set_id	ue(v)

croma_format_idc	ue(v)
...	
log2_max_pic_order_cnt_lsb_minus4	ue(v)
poc_msb_in_rap_pics_flag	u(1)
si (poc_msb_in_rap_pics_flag > 0)	
poc_msb_len_minus1	ue(v)
si (sps_max_sub_layers_minus1 > 0))	
sps_sub_layer_dpb_params_flag	u(1)
si (sps_ptl_dpb_present flag)	
dpb_parameters(0, sps_max_sub_layers_minus1, sps_sub_layer_dpb_params_flag)	
para (i = (sps_sub_layer_ordering_info_present_flag? 0: sps_max_sub_layers_minus1);	
i <= sps_max_sub_layers_minus1; i++) {	
sps_max_dec_pic_buffering_minus1[i]	ue(v)
sps_max_num_reorder_pics[i]	ue(v)
sps_max_latency_increase_plus1[i]	ue(v)
}	
long_term_ref_pics_flag	u(1)
...	
sps_scaling_list_enabled_flag	u(1)
general_hrd_parameters_present_flag	u(1)
si (general_hrd_parameters_present_flag) {	
num_units_in_tick	u(32)
time_scale	u(32)
sub_layer_cpb_parameters_present_flag	u(1)
si (sub_layer_cpb_parameters_present_flag)	
general_hrd_parameters(0, sps_max_sub_layers_minus1)	
si no	
general_hrd_parameters(sps_max_sub_layers_minus1, sps_max_sub_layers_minus1)	
}	
vui_parameters_present_flag	u(1)
si (vui_parameters_present_flag)	
vui_parameters( )	
sps_extension_flag	u(1)
si (sps_extension_flag)	
mientras (more_rbsp_data ( ))	
sps_extension_data_flag	u(1)
rbsp_trailing_bits( )	
}	

Un ejemplo de sintaxis de parámetros de DPB es el siguiente.

dpb_parameters(dpbSizeOnlyFlag, maxSubLayersMinus1, subLayerInfoFlag) {	Descriptor
para (i = (subLayerInfoFlag? 0: maxSubLayersMinus1);	
i <= maxSubLayersMinus1; i++) {	
max_dec_pic_buffering_minus1[i]	ue(v)
si (!dpbSizeOnlyFlag) {	
max_num_reorder_pics[i]	ue(v)

max latency_increase_plus1[i]	ue(v)
}	
}	
}	

Un ejemplo de sintaxis general de parámetros de HRD es el siguiente.

general_hrd_parameters( )	Descriptor
general_nal_hrd_params_present_flag	u(1)
general_vcl_hrd_params_present_flag	u(1)
si (general_nal_hrd_params_present_flag | | general _vcl_hrd_params_present_flag) {	
decoding_unit_hrd_params_present_flag	u(1)
si (decoding_unit_hrd_params_present _flag) {	
tick_divisor_minus2	u(8)
decoding_unit_cpb_params_in_pic_timing_sei_flag	u(1)
}	
bit_rate_scale	u(4)
cpb_size_scale	u(4)
si (decoding_unit_hrd_params_present_flag)	
cpb_size_du_scale	u(4)
}	
si (vps_max_sub_layers_minus1 > 0)	
sub_layer_cpb_params_present_flag	u(1)
si (TotalNumOlss > 1)	
num_ols_hrd_params_minus1	ue(v)
hrd_cpb_cnt_minus1	ue(v)
para (i = 0; i <= num_ols_hrd_params_minus1; i++) {	
si (vps_max_sub_layers_minus1 > 0 && !vps_all_layers_same_num_sub_layers_flag)	
hrd_max_temporal_id[i]	u(3)
ols_hrd_parameters(hrd_max_temporal_id[i])	
}	
si (num_ols_hrd_params_minus1 > 0)	
para(i = 1; i < TotalNumOlss; i++)	
ols_hrd_idx[i]	ue(v)
}	

Un ejemplo de sintaxis de parámetros de HRD de OLS es el siguiente.

ols_hrd_parameters(hrdMaxTid) {	Descriptor
firstSubLayer = sub_layer_cpb_params_present_flag? 0: hrdMaxTid	
para(i = firstSubLayer; i <= hrdMaxTid; i++) {	
fix_pic_rate_general_flag[i]	u(1)
si (!fixed_pic_rate_general_flag[i])	
fixed_pic_rate_within_cvs_flag[i]	u(1)
si (fixed_pic_rate_within_cvs_flag[i])	
elemental_duration_in_tc_minus1[i]	ue(v)
si no, si (hrd_cpb_cnt_minus1 = = 0)	
low_delay_hrd_flag[i]	u(1)

si (general_nal_hrd_params_present_ flag)	
sublayer_hrd_parameters (i)	
si (general_vcl_hrd_params_present_flag)	
sub_layer_hrd_parameters(i)	
}	
}	

Un ejemplo de sintaxis de parámetros de HRD de subcapa es el siguiente.

sub_layer_hrd_parameters (subLayerId) {	Descriptor
para(j = 0; j <= hrd_cpb_cnt_minus1;j++) {	
bit_rate_value_minus1[subLayerId] [j]	ue(v)
cpb_size_value_minus1[subLayerId][j]	ue(v)
si (decoding_unit_hrd_params_present_flag)	
cpb_size_value_minus1[subLayerId][j]	ue(v)
bit_rate_value_minus1[subLayerId] [j]	ue(v)
}	
cbr_flag[subLayerId][j]	u(1)
}	
}	

Un ejemplo de semántica de RBSP de conjunto de parámetros de vídeo es el siguiente. Un vps_max_layers_minus1 + 1 especifica el número máximo permitido de capas en cada CVS en referencia al VPS. Un vps_max_sublayers_minus1 + 1 especifica el número máximo de subcapas temporales que pueden estar presentes en cada CVS en referencia al VPS. El valor de vps_max_sub_layers_minus1 puede estar en el rango de cero a seis, inclusive. Un vps_all_layers_same_num_sub_layers_flag igual a uno especifica que el número de subcapas temporales es el mismo para todas las capas en cada CVS en referencia al VPS. Un vps_all_layers_same_num_sub_layers_flag igual a cero especifica que las capas en cada CVS que hace referencia al VPS pueden tener o no el mismo número de subcapas temporales. Cuando no está presente, se puede inferir que el valor de vps_all_layers_same_num_sub_layers_flag es igual a uno. Un vps_all_independent_layers_flag igual a uno especifica que todas las capas en el CVS están codificadas de manera independiente sin utilizar predicción entre capas. Un vps_all_independent_layers_flag igual a cero especifica que una o más de las capas en el CVS pueden usar predicción entre capas. Cuando no está presente, se puede inferir que el valor de vps_all_independent_layers_flag es igual a uno. cuando el vps_all_independent_layers_flag es igual a uno, se infiere que el valor de vps_independent_layer_flag[i] es igual a uno. cuando el vps_all_independent_layers_flag es igual a cero, se infiere que el valor de vps_independent_layer_flag[0] es igual a uno.

Un vps_direct_dependency_flag[i][j] igual a cero especifica que la capa con índice j no es una capa de referencia directa para la capa con índice i. Un vps_direct_dependency_flag[i][j] igual a uno especifica que la capa con índice j es una capa de referencia directa para la capa con índice i. cuando el vps_direct_dependency_flag[i][j] no está presente para i y j en el rango de cero a vps_max_layers_minus1, inclusive, se infiere que el indicador es igual a cero. La variable DirectDependentLayerIdx[i][j], que especifica la capa de orden j dependiente directa de la capa de orden i, y la variable LayerUsedAsRefLayerFlag[j], que especifica si la capa con índice de capa j se utiliza como capa de referencia por cualquier otra capa, se puede obtener de la siguiente manera:

```
para( i = 0; i <= vps_max_layers_minus1; i++ )
        LayerUsedAsRefLayerFlag[ j ] = 0
para( i = 1; i < vps_max_layers_minus1; i++ )
        si( !vps_independent_layer_flag[ i ] )
                para ( j = i − 1, k = 0; j >= 0; j− − )
                        si( vps_direct_dependency_flag[ i ][ j ] ) {
                                DirectDependentLayerIdx[ i ][ k++ ] = j
                                LayerUsedAsRefLayerFlag[ j ] = 1
                        }
```

La variable GeneralLayerIdx[i], que especifica el índice de capa de la capa con nuh_layer_Id igual a vps_layer_id[i], se puede obtener de la siguiente manera:

para(i = 0; i <= vps_max_layers_minus1; i++) GeneralLayerIdx[vps_layer_id[i]]= i

Un each_layer_is_an_ols_flag igual a uno, especifica que cada conjunto de capas de salida contiene solo una capa, y cada capa en sí en el flujo de bits es un conjunto de capas de salida, siendo la única capa incluida la única capa de salida. Un each_layer_is_an_ols_flag igual a cero especifica que un conjunto de capas de salida puede contener más de una capa. Si vps_max_layers_minus1 es igual a cero, se infiere que el valor de each_layer_is_an_ols_flag es igual a uno. De lo contrario, cuando el vps_all_independent_layers_flag es igual a cero, se infiere que el valor de each_layer_is_an_ols_flag es igual a cero.

Un ols_mode_idc igual a cero especifica que el número total de OLS especificados por el VPS es igual a vps_max_layers_minus1 + 1, el OLS de orden i incluye las capas con índices de capa de cero a i, inclusive, y para cada OLS solo se genera la capa más alta en el OLS. Un ols_mode_idc igual a uno especifica que el número total de OLS especificados por el VPS es igual a vps_max_layers_minus1 + 1, el OLS de orden i incluye las capas con índices de capa de cero a i, inclusive, y para cada OLS se generan todas las capas del OLS. Un ols_mode_idc igual a dos especifica que el número total de OLS especificados por el VPS se señaliza explícitamente y para cada OLS se generan la capa más alta y un conjunto de capas inferiores explícitamente señalizadas en el OLS. El valor de ols_mode_idc puede estar en un rango de cero a dos, inclusive. cuando el vps_all_independent layers_flag es igual a uno e each_layer_is_an_ols_flag es igual a cero, se infiere que el valor de ols_mode_idc es igual a dos. Un num_output_layer_sets_minus1 + 1 especifica el número total de OLS especificados por el VPS cuando ols_mode_idc es igual a dos.

La variable TotalNumOlss, que especifica el número total de OLS especificados por el VPS, se puede obtener de la siguiente manera:

```
si( vps_max_layers_minus1 == 0 )
        TotalNumOlss = 1
si no, si( each_layer_is_an_ols_flag || ols_mode_idc == 0 || ols_mode_idc == 1 )
        TotalNumOlss = vps_max_layers_minus1 + 1
si no, si( ols_mode_idc == 2 )
        TotalNumOlss = num_output_layer_sets_minus1 + 1
```

Un layer_included_flag[i][j] especifica si la capa de orden j (por ejemplo, la capa con nuh_layer_id igual a vps_layer_id[j]) está incluida en el OLS de orden i cuando ols_mode_idc es igual a dos. Un layer_included_flag[i][j] igual a uno especifica que la capa de orden j está incluida en el OLS de orden i. Un layer_included_flag[i][j] igual a cero especifica que la capa de orden j no está incluida en el OLS de orden i. La variable NumLayersInOls[i], que especifica el número de capas en el OLS de orden i, y la variable LayerIdInOls[i][j], que especifica el valor de nuh_layer_id de la capa de orden j en el OLS de orden i, pueden obtenerse de la siguiente manera:

```
NumLayersInOls[ 0 ] = 1
LayerIdInOls[ 0 ][ 0 ] = vps_layer_id[ 0 ]
para( i = 1, i < TotalNumOlss; i++ ) {
        si ( each_layer_is_an_ols_flag ) {
                NumLayersInOls[ i ] = 1
                LayerIdInOls[ i ][ 0 ] = vps_layer_id[ i ]
        } si no, si( ols_mode_idc == 0 || ols_mode_idc == 1 ) {
                NumLayersInOls[ i ] = i + 1
                para( j = 0; j < NumLayersInOls[ i ]; j++ )
                        LayerIdInOls[ i ][ j ] = vps_layer_id[ j ]
        } si no, si( ols_mode_idc == 2 ) {
                para ( k = 0, j = 0; k <= vps_max_layers_minus1; k++ )
                        si( layer_included_flag[ i ][ k ] )
                                LayerIdInOls[ i ][ j++ ] = vps_layer_id[ k ]
                NumLayersInOls[ i ] = j
        }
}
```

La variable OlsLayerIdx[i][j], que especifica el índice de capa OLS de la capa con nuh_layer_id igual a LayerIdInOls[i][j], se puede obtener de la siguiente manera:

```
para( i = 0, i < TotalNumOlss; i++ )
        para j = 0; j < NumLayersInOls[ i ]; j++ )
                OlsLayeIdx[ i ][ LayerIdInOls[ i ][ j ] ] = j
```

La capa más baja en cada OLS será una capa independiente. En otras palabras, para cada i en el rango de cero a TotalNumOlss - 1, inclusive, el valor de vps_independent_layer_flag[GeneralLayerIdx[LayerIdInOls[i][0]]] será igual a uno. Cada capa debe estar incluida en al menos un OLS especificado por el VPS. En otras palabras, para cada capa con un valor particular de id de capa de nuh, nuh_layer_id, igual a uno de vps_layer_id[k] para k en el rango de cero a vps_max_layers_minus1, inclusive, habrá al menos un par de valores de i y j, donde i está en el rango de cero a TotalNumOlss - 1, inclusive, y j está en el rango de NumLayersInOls[i] - 1, inclusive, de tal manera que el valor de LayerIdInOls[i][j] es igual a nuh_layer_Id. Cualquier capa de un OLS será una capa de salida del OLS o una capa de referencia (directa o indirecta) de una capa de salida del OLS.

Un vps_output_layer_flag[i][j] especifica si la capa de orden j en el OLS de orden i se genera cuando ols_mode_idc es igual a dos. Un vps_output_layer_flag[i] igual a uno especifica que se genera la capa de orden j en el OLS de orden i. Un vps_output_layer_flag[i] igual a cero especifica que no se genera la capa de orden j en el OLS de orden i. Cuando vps_all_independent_layers_flag es igual a uno y each_layer_is_an_ols_flag es igual a cero, se infiere que el valor de vps_output_layer_flag[i] es igual a uno. La variable OutputLayerFlag[i][j], para la cual el valor uno especifica que se genera la capa de orden j en el OLS de orden i, y el valor cero especifica que la capa de orden j en el OLS de orden i no se genera, se puede obtener de la siguiente manera.

```
para( i = 0; i < TotalNumOlss; i++ ) {
        OutputLayerFlag[ i ][ NumLayersInOls[ i ] − 1 ] = 1
    para( j = 0; j < NumLayersInOls[ i ] − 1; j++ )
        si( ols_mode_idc[ i ] == 0 )
            OutputLayerFlag[ i ][ j ] = 0
        si no, si( ols_mode_idc[ i ] == 1 )
            OutputLayerFlag[ i ][ j ] = 1
        si no, si( ols_mode_idc[ i ] == 2 )
            OutputLayerFlag[ i ][ j ] = vps_output_layer_flag[ i ][ j ]
}
```

Cabe señalar que un OLS de orden 0 contiene solo la capa más baja (por ejemplo, la capa con nuh_layer_id igual a vps_layer_id[0]) y para el OLS de orden 0 se genera la única capa incluida. Un vps_num_ptls especifica el número de estructuras de sintaxis de profile_tier_level( ) en el VPS. Un pt_present_flag[i] igual a uno especifica que la información de perfil, nivel y restricciones generales están presentes en la estructura de sintaxis profile_tier_level( ) de orden i en el VPS. Un pt_present_flag[i] igual a cero especifica que la información de perfil, nivel y restricciones generales no está presente en la estructura de sintaxis profile_tier_level( ) de orden i en el VPS. Se infiere que el valor de pt_present_flag[0] es igual a cero. Cuando pt_present _flag[i] es igual a cero, se infiere que la información de perfil, nivel y restricciones generales para la estructura de sintaxis profile_tier_level( ) de orden i en el VPS es la misma que para la estructura de sintaxis profile_tier_level( ) de orden (i-1) en el VPS.

Un ptl_max_temporal_id[i] especifica el TemporalId de la representación de subcapa más alta para la cual la información de nivel está presente en la estructura de sintaxis profile_tier_level( ) de orden i en el VPS. El valor de ptl_max_temporal_id[i] estará en el rango de cero a vps_max_sub_layers_minus1, inclusive. cuando el vps_max_sub_layers_minus1 es igual a cero, se infiere que el valor de ptl_max_temporal_id[i] es igual a cero. cuando el vps_max_sub_layers_minus1 es mayor que cero y vps_all_layers_same_num_sub_layers_flag es igual a uno, se infiere que el valor de ptl_max_temporal_id[i] es igual a vps_max_sub_layers_minus1. Un vps_ptl_byte_alignment_zero_bit debe ser igual a cero.

Un ols_ptl_idx[i] especifica el índice, a la lista de estructuras de sintaxis profile_tier_level( ) en el VPS, de la estructura de sintaxis profile_tier_level( ) que se aplica al OLS de orden i. Cuando está presente, el valor de ols_ptl_idx[i] debe estar en el rango de cero a vps_num_ptls - 1, inclusive. Cuando NumLayersInOls[i] es igual a uno, la estructura de sintaxis profile_tier_level( ) que se aplica al OLS de orden i está presente en el SPS al que hace referencia la capa en el OLS de orden i. Un vps_num_dpb_params especifica el número de estructuras de sintaxis dpb_parameters( ) en el VPS. El valor de vps_num_dpb_params estará en el rango de cero a dieciséis, inclusive. Cuando no está presente, se puede inferir que el valor de vps_num_dpb_params es igual a cero. Un same_dpb_size_output_or_nonoutput_flag igual a uno especifica que no hay ningún elemento de sintaxis layer_nonoutput_dpb_params_idx[i] presente en el VPS. Un same_dpb_size_output_or_nonoutput_flag igual a uno especifica que no hay ningún elemento de sintaxis layer_nonoutput_dpb_params_idx[i] presente en el VPS. Se utiliza un vps_sub_layer_dpb_params_present_flag para controlar la presencia de los elementos de sintaxis max_dec_pic_buffering_minus1[ ], max_num_reorder_pics[ ] y max_latency_increase_plus1[ ] en las estructuras de sintaxis dpb_parameters( ) en el VPS. Cuando no está presente, se infiere que vps_sub_dpb_params_info_present_flag es igual a cero.

Un dpb_size_only_flag[i] igual a uno especifica que los elementos de sintaxis max_num_reorder_pics[ ] y max_latency_increase_plus1[ ] no están presentes en las estructuras de sintaxis de orden i dpb_parameters( ) del VPS. Un dpb_size_only_flag[i] igual a uno especifica que los elementos de sintaxis max_num_reorder_pics[ ] y max_latency_increase_plus1[ ] no están presentes en las estructuras de sintaxis de orden i dpb_parameters( ) del VPS. Un dpb_max_temporal_id[i] especifica el TemporalId de la representación de subcapa más alta para la cual los parámetros de DPB pueden estar presentes en la estructura de sintaxis dpb_parameters( ) de orden i en el VPS. El valor de dpb_max_temporal_id[i] debe estar en el rango de cero a vps_max_sub_layers_minus1, inclusive. cuando el vps_max_sub_layers_minus1 es igual a cero, se puede inferir que el valor de dpb_max_temporal_id[i] es igual a cero. cuando el

vps_max_sub_layers_minus1 es mayor que cero y vps_all_layers_same_num_sub_layers_flag es igual a uno, se infiere que el valor de dpb_max_temporal_id[i] es igual a vps_max_sub_layers_minus1. Un layer_output_dpb_params_idx[i] especifica el índice, a la lista de estructuras de sintaxis dpb_parameters( ) en el VPS, de la estructura de sintaxis dpb_parameters( ) que se aplica a la capa de orden i cuando es una capa de salida en un OLS. Cuando esté presente, el valor de layer_output_dpb_params_idx[i] deberá estar en el rango de cero a vps_num_dpb_params - 1, inclusive.

Si vps_independent_layer_flag[i] es igual a uno, la estructura de sintaxis dpb_parameters( ) que se aplica a la capa de orden i cuando es una capa de salida es la estructura de sintaxis dpb_parameters( ) presente en el SPS mencionado por la capa. De lo contrario (vps_independent_layer_flag[i] es igual a uno), se aplica lo siguiente. cuando el vps_num_dpb_params es igual a uno, se infiere que el valor de layer_output_dpb_params_idx[i] es igual a cero. La conformidad del flujo de bits puede requerir que el valor de layer_output_dpb_params_idx[i] sea tal que dpb_size_only_flag[layer_output_dpb_params_idx[i]] sea igual a cero.

Un layer_nonoutput_dpb_params_idx[i] especifica el índice, a la lista de estructuras de sintaxis dpb_parameters( ) en el VPS, de la estructura de sintaxis dpb_parameters( ) que se aplica a la capa de orden i cuando es una capa no de salida en un OLS. Cuando esté presente, el valor de layer_nonoutput_dpb_params_idx[i] deberá estar en el rango de cero a vps_num_dpb_params - 1, inclusive. Si same_dpb_size_output_or_nonoutput_flag es igual a 1, se aplica lo siguiente. Si vps_independent_layer_flag[i] es igual a uno, la estructura de sintaxis dpb_parameters( ) que se aplica a la capa de orden i cuando es una capa de salida es la estructura de sintaxis dpb_parameters( ) presente en el SPS mencionado por la capa. De lo contrario (vps_independent_layer_flag[i] es igual a uno), se infiere que el valor de Layer_nonoutput_dpb_params_idx[i] es igual a Layer_output_dpb_params_idx[i]. De lo contrario (same_dpb_size_output_or_nonoutput_flag es igual a cero), cuando el vps_num_dpb_params es igual a uno, se infiere que el valor de layer_output_dpb_params_idx[i] es igual a cero.

Un general_hrd_params_present_flag igual a uno especifica que los elementos de sintaxis num_units_in_tick (número de unidades en tictac) y time_scale (escala de tiempo) y la estructura de sintaxis general_hrd_parameters( ) están presentes en la estructura de sintaxis de RBSP de SPS. Un general_hrd_params_present_flag igual a cero especifica que los elementos de sintaxis num_units_in_tick y time_scale y la estructura de sintaxis general_hrd_parameters( ) no están presentes en la estructura de sintaxis de RBSP de SPS. Un num_units_in_tick es el número de unidades de tiempo de un reloj que funciona en la frecuencia de time_scale hercios (Hz) que corresponde a un incremento (llamado tictac de reloj) de un contador de tictacs de reloj. Un num_units_in_tick será mayor que cero. Un tictac de reloj, en unidades de segundos, es igual al cociente de num_units_in_tick dividido por time_scale. Por ejemplo, cuando la velocidad de imagen de una señal de vídeo es veinticinco Hz, time_scale puede ser igual a 27.000.000 y num_units_in_tick puede ser igual a 1.080.000 y, en consecuencia, un tictac de reloj puede ser igual a 0,04 segundos. Un time_scale es el número de unidades de tiempo que pasan en un segundo. Por ejemplo, un sistema de coordenadas de tiempo que mide el tiempo usando un reloj de 27 MHz tiene un time_scale de 27.000.000. El valor de time_scale será mayor que cero.

Un vps_extension_flag igual a cero especifica que no hay elementos de sintaxis vps_extension_data_flag presentes en la estructura de sintaxis de RBSP de VPS. Un vps_extension_flag igual a uno especifica que hay elementos de sintaxis vps_extension_data_flag presentes en la estructura de sintaxis de RBSP de VPS. Un vps_extension_data_flag puede tener cualquier valor. Es posible que la presencia y el valor de vps_extension_data_flag no afecten a la conformidad del decodificador con los perfiles. Los decodificadores conformes pueden ignorar todos los elementos de sintaxis vps_extension_data_flag.

Un ejemplo de semántica de RBSP de conjunto de parámetros de secuencia es el siguiente. Un RBSP de SPS debe estar disponible para el proceso de decodificación antes de ser referenciado, incluido en al menos una unidad de acceso con TemporalId igual a cero, o proporcionado a través de medios externos, y la unidad de NAL de SPS que contiene el RBSP de SPS debe tener un nuh_layer_id igual al valor de nuh_layer_id más bajo de las unidades de NAL de PPS que se refieren a la unidad de NAL de SPS. Todas las unidades de NAL de SPS con un valor particular de sps_seq_parameter_set_id en un CVS deben tener el mismo contenido. Un sps_decoding_parameter_set_id, cuando es mayor que cero, especifica el valor de dps_decoding_parameter_set_id para el DPS al que hace referencia el SPS. cuando el sps_decoding_parameter_set_id es igual a cero, el SPS no hace referencia a un DPS y no se hace referencia a ningún DPS al decodificar cada CLVS que hace referencia al SPS. El valor de sps_decoding_parameter_set_id debe ser el mismo en todos los SPS a los que hacen referencia las imágenes codificadas en un flujo de bits.

Un sps_video_parameter_set_id, cuando es mayor que cero, especifica el valor de vps_video_parameter_set_id para el VPS al que hace referencia el SPS. cuando el sps_video_parameter_set_id es igual a cero, el SPS puede no hacer referencia a un VPS y no se hace referencia a ningún VPS al decodificar cada CLVS que hace referencia al SPS, y se debe inferir que el valor

de GeneralLayerIdx[nuh_layer_id] es igual a cero, y se puede inferior que el valor de vps_independent_layer_flag[GeneralLayerIdx[nuh_layer_id]] es igual a uno. cuando el vps_independent_layer_flag[GeneralLayerIdx[nuh_layer_id]] es igual a uno, el SPS al que hace referencia un CLVS con un valor de nuh_layer_id particular nuhLayerId tendrá un nuh_layer_id igual a nuhLayerId.

Un sps_max_sub_layers_minus1 + 1 especifica el número máximo de subcapas temporales que pueden estar presentes en cada CVS al que hace referencia el SPS. El valor de sps_max_sub_layers_minus1 debe estar en el rango de cero a vps_max_sub_layers_minus1, inclusive. Un sps_reserved_zero_4bits debe ser igual a cero en los flujos de bits conformes. Se pueden reservar otros valores para sps_reserved_zero_4bits.

Un sps_ptl_dpb_present_flag igual a uno especifica que una estructura de sintaxis profile_tier_level( ) y una estructura de sintaxis dpb_parameters( ) están presentes en el SPS. Un sps_ptl_dpb_present_flag igual a cero especifica que una estructura de sintaxis profile_tier_level( ) y una estructura de sintaxis dpb_parameters( ) están presentes en el SPS. El valor de sps_ptl_dpb_present_flag debe ser igual a vps_independent_layer_flag[nuh_layer_id]. Si vps_independent_layer_flag[GeneralLayerIdx[nuh_layer_id]] es igual a uno, la variable MaxDecPicBuffMinus1 se ajusta igual a max_dec_pic_buffering_minus1[sps_max_sub_layers_minus1] en la estructura de sintaxis dpb_parameters( ) en el SPS. De lo contrario, MaxDecPicBuffMinus1 se ajusta igual a max_dec_pic_buffering_minus1[sps_max_sub_layers_minus1] en la estructura de sintaxis layer_nonoutput_dpb_params_idx[GeneralLayerIdx[nuh_layer_id]] - la estructura de sintaxis dpb_parameters( ) en el VPS. Un gdr_enabled_flag igual a uno especifica que las imágenes de GDR pueden estar presentes en los CLVS que hacen referencia al SPS. Un gdr_enabled_flag igual a cero especifica que las imágenes de GDR pueden estar presentes en los CLVS que hacen referencia al SPS.

Se utiliza un sps_sub_layer_dpb_params_present_flag para controlar la presencia de elementos de sintaxis max_dec_pic_buffering_minus1[i], max_num_reorder_pics[i] y max_latency_increase_plus1[i] en las estructuras de sintaxis de dpb_parameters( ) en el SPS. Cuando no está presente, se infiere que sps_sub_dpb_params_info_present_flag es igual a cero. Un long_term_ref_pics_flag igual a cero especifica que no se utiliza ningún LTRP para la interpredicción de ninguna imagen codificada en el CLVS. Un long_term_ref_pics_flag igual a uno especifica que los LTRP se pueden utilizar para la interpredicción de una o más imágenes codificadas en el CLVS.

Un ejemplo de semántica general de perfil, escalonado y nivel es el siguiente. Una estructura de sintaxis profile_tier_level( ) proporciona información de nivel y, opcionalmente, información de perfil, nivel, subperfil e información de restricciones generales (indicada como información de PT). Cuando la estructura de sintaxis profile_tier_level( ) se incluye en un DPS, OlsInScope es el OLS que incluye todas las capas en todo el flujo de bits que hace referencia al DPS. Cuando la estructura de sintaxis profile_tier_level( ) se incluye en un VPS, OlsInScope es uno o más OLS especificados por el VPS. Cuando la estructura de sintaxis profile_tier_level( ) se incluye en un SPS, OlsInScope es el OLS que incluye solo la capa que es la capa más baja entre las capas que hacen referencia al SPS, que debe ser una capa independiente.

Un general_profile_idc indica un perfil al que se ajusta OlsInScope. Un indicador de nivel general especifica el contexto de nivel para la interpretación de general_level_idc. Un num_sub_profiles especifica el número de elementos de sintaxis general_sub_profile_idc[i]. Un general_sub_profile_idc[i] indica el metadato de interoperabilidad de orden i registrado. Un general_level_idc indica un nivel al que se ajusta OlsInScope. Cabe señalar que un valor mayor de general_level_idc indica un nivel más alto. El nivel máximo señalizado en el DPS para OlsInScope puede ser mayor que el nivel señalado en el SPS para un CVS contenido en OlsInScope. También cabe señalar que cuando OlsInScope se ajusta a múltiples perfiles, general_profile_idc debe indicar el perfil que proporciona el resultado decodificado preferido o la identificación de flujo de bits preferida, según lo determine el codificador. También se debe tener en cuenta que cuando la estructura de sintaxis profile_tier_level( ) se incluye en un DPS y los CVS de OlsInScope se ajustan a diferentes perfiles, general_profile_idc y nivel_idc deben indicar el perfil y el nivel de un decodificador que es capaz de decodificar OlsInScope.

Un sublayer_level_present_flag[i] igual a uno especifica que la información de nivel está presente en la estructura de sintaxis profile_tier_level( ) para la representación de subcapa con TemporalId igual a i. Un sublayer_level_present_flag[i] igual a cero especifica que la información de nivel no está presente en la estructura de sintaxis profile_tier_level( ) para la representación de subcapa con TempralId igual a i. Un ptl_alignment_zero_bits debe ser igual a cero. La semántica del elemento de sintaxis sub_layer_level_idc[i] es, aparte de la especificación de la inferencia de valores no presentes, la misma que la del elemento de sintaxis general_level_idc, pero se aplica a la representación de subcapa con TemporalId igual a i.

Un ejemplo de semántica de parámetros de DPB es el siguiente. La estructura de sintaxis dpb_parameters(maxSubLayersMinus1, subLayerInfoFlag) proporciona información sobre el tamaño de DPB, el número máximo de reordenamiento de imágenes y la latencia máxima para cada CLVS del CVS. Cuando se incluye una estructura de sintaxis dpb_parameters( ) en un VPS, el VPS especifica los OLS a los que se

aplica la estructura de sintaxis dpb_parameters( ). Cuando se incluye una estructura de sintaxis dpb_parameters( ) en un SPS, la estructura de sintaxis dpb_parameters( ) se aplica al OLS que incluye solo la capa que es la capa más baja entre las capas que hacen referencia al SPS, que será una capa independiente.

Un max_dec_pic_buffering_minus1[i] + 1 especifica, para cada CLVS del CVS, el tamaño máximo requerido de la memoria intermedia de imágenes decodificadas en unidades de memorias intermedias de almacenamiento de imágenes cuando Htid es igual a i. El valor de max_dec_pic_buffering_minus1[i] debe estar en el rango de 0 a MaxDpbSize - 1, inclusive. Cuando i es mayor que cero, max_dec_pic_buffering_minus1[i] debe ser mayor o igual que max_dec_pic_buffering_minus1[i - 1]. Cuando max_dec_pic_buffering_minus1[i] no está presente para i en el rango de cero a maxSubLayersMinus1 - 1, inclusive, debido a que subLayerInfoFlag es igual a cero, se infiere que max_dec_pic_buffering_minus1[i] es igual a max_dec_pic_buffering_minus1[maxSubLayersMinus1].

Un max_num_reorder_pics[i] especifica, para cada CLVS del CVS, el número máximo permitido de imágenes del CLVS que pueden preceder a cualquier imagen en el CLVS en orden de decodificación y seguir a esa imagen en orden de salida cuando Htid es igual a i. El valor de max_num_reorder_pics[i] debe estar en el rango de cero a max_dec_pic_buffering_minus1[i], inclusive. Cuando i es mayor que cero, max_num_reorder_pics[i] debe ser mayor o igual que max_num_reorder_pics[i - 1]. Cuando max_num_reorder_pics[i] no está presente para i en el rango de cero a maxSubLayersMinus1 - 1, inclusive, debido a que subLayerInfoFlag es igual a cero, se infiere que max_num_reorder_pics[i] es igual a max_num_reorder_pics[maxSubLayersMinus1].

Se utiliza un max_latency_increase_plus1[i] distinto de cero para calcular el valor de MaxLatencyPictures[i], que especifica, para cada CLVS del CVS, el número máximo de imágenes en el CLVS que pueden preceder a cualquier imagen en el CLVS en orden de salida y seguir a esa imagen en orden de decodificación cuando Htid es igual a i. Cuando max_latency_increase_plus1[i] no es igual a cero, el valor de MaxLatencyPictures[i] se puede especificar de la siguiente manera.

$$MaxLatencyPictures[i] = max\_num\_reorder\_pics[i] + max\_latency\_increase\_plus1[i] - 1$$

Cuando max_latency_increase_plus1[i] es igual a 0, no se expresa ningún límite correspondiente.

El valor de max_latency_increase_plus1[i] debe estar en el rango de cero a doscientos treinta y dos menos dos, inclusive. Cuando max_latency_increase_plus1[i] no está presente para i en el rango de cero a maxSubLayersMinus1 - 1, inclusive, debido a que subLayerInfoFlag es igual a cero, se infiere que max_latency_increase_plus1[i] es igual a max_latency_increase_plus1[maxSubLayersMinus1].

Un ejemplo de semántica general de parámetros de HRD es el siguiente. La estructura de sintaxis general_hrd_parameters( ) proporciona los parámetros de HRD utilizados en las operaciones de HRD. Un num_ols_hrd_params_minus1 + 1 especifica el número de estructuras de sintaxis ols_hrd_parameters( ) presentes en la estructura de sintaxis general_hrd_parameters( ). El valor de num_ols_hrd_params_minus1 debe estar en el rango de cero a sesenta y tres, inclusive. Cuando TotalNumOlss es mayor que uno, se infiere que el valor de num_ols_hrd_params_minus1 es igual a cero. El hrd_cpb_cnt_minus1 + 1 especifica el número de especificaciones de CPB alternativas en el flujo de bits del CVS. El valor de hrd_cpb_cnt_minus1 debe estar en el rango de cero a treinta y uno, inclusive. Un hrd_max_temporal_id[i] especifica el TemporalId de la representación de subcapa más alta para la cual los parámetros de HRD están contenidos en la estructura de sintaxis layer_level_hrd_parameters( ) de orden i. El valor de hrd_max_temporal_id[i] debe estar en el rango de cero a vps_max_sub_layers_minus1, inclusive. cuando el vps_max_sub_layers_minus1 es igual a cero, se infiere que el valor de hrd_max_temporal_id[i] es igual a cero. Un ols_hrd_idx[i] especifica el índice de la estructura de sintaxis ols_hrd_parameters( ) que se aplica al OLS de orden i. El valor de ols_hrd_idx[[i] debe estar en el rango de cero a num_ols_hrd_params_minus1, inclusive. Cuando no está presente, se infiere que el valor de ols_hrd_idx[[i] es igual a cero.

Un ejemplo de semántica de estructura de lista de imágenes de referencia es el siguiente. La estructura de sintaxis ref_pic_list_struct(listIdx, rplsIdx) puede estar presente en un SPS o en una cabecera de segmento. Dependiendo de si la estructura de sintaxis se incluye en una cabecera de segmento o en un SPS, se aplica lo siguiente. Si está presente en una cabecera de segmento, la estructura de sintaxis ref_pic_list_struct(listIdx, rplsIdx) especifica la lista de imágenes de referencia listIdx de la imagen actual (la imagen que contiene el segmento). De lo contrario (presente en un SPS), la estructura de sintaxis ref_pic_list_struct(listIdx, rplsIdx) especifica un candidato para la lista de imágenes de referencia listIdx, y el término imagen actual en la semántica especificada en el resto de este apartado se refiere a cada imagen que tiene uno o más segmentos que contienen ref_pic_list_idx[listIdx] equivalen a un índice en la lista de estructuras de sintaxis ref_pic_list_struct(listIdx, rplsIdx) incluidas en el SPS, y están en un CVS que hace referencia al SPS. Un num_ref_entries[listIdx][rplsIdx] especifica el número de entradas en la estructura de sintaxis ref_piclist_struct(listIdx, rplsIdx). El valor de num_ref_entries[listIdx][rplsIdx] debe estar en el rango de cero a MaxDecPicBuffMinus1 + catorce, inclusive.

Un ejemplo de proceso de decodificación general es el siguiente. La entrada a este proceso es un flujo de bits BitstreamToDecode (flujo de bits a decodificar). El resultado de este proceso es una lista de imágenes decodificadas. El proceso de decodificación se especifica de tal manera que todos los decodificadores que se ajustan a un perfil y nivel especificados producen imágenes de salida decodificadas recortadas, numéricamente idénticas, al invocar el proceso de decodificación asociado con ese perfil para un flujo de bits conforme a ese perfil y nivel Cualquier proceso de decodificación que produzca imágenes de salida decodificadas recortadas, idénticas a las producidas por el proceso descrito en este documento (con el orden de salida correcto o el tiempo de salida correcto, según se especifica), se ajusta a los requisitos del proceso de decodificación.

Para cada AU de IRAP en el flujo de bits, se aplica lo siguiente. Si la AU es la primera AU en el flujo de bits en orden de decodificación, cada imagen es una imagen de actualización de decodificación instantánea (Instantaneous Decoding Refresh, IDR), o cada imagen es la primera imagen de la capa que sigue a una unidad de NAL de fin de secuencia en orden de decodificación, la variable NoIncorrectPicOutputFlag se ajusta a uno. De lo contrario, si la variable HandleCraAsCbsStattFlag se ajusta a un valor para la AU, HandleCraAsCvsStartFlag se ajusta a un valor proporcionado por un mecanismo externo y NoIncorrectPicOutputFlag se ajusta a HandleCraAsCvsStartFlag. De lo contrario, HandleCraAsCbsStattFlag y NoIncorrectPicOutputFlag se ajustan en cero.

Para cada AU de actualización de decodificación gradual (Gradual Decoding Refresh, GDR) en el flujo de bits, se aplica lo siguiente. Si la AU es la primera AU en el flujo de bits en orden de decodificación o cada imagen es la primera imagen de la capa que sigue a un final de unidad de NAL de secuencia en orden de decodificación, la variable NoIncorrectPicOutputFlag se ajusta a uno. De lo contrario, si hay algún mecanismo externo disponible para ajustar la variable HandleGdrAsCvsStartFlag a un valor para la AU, HandleGdrAsCvsStartFlag se ajusta al valor proporcionado por el mecanismo externo y NoIncorrectPicOutputFlag se ajusta a HandleGdrAsCvsStartFlag. De lo contrario, HandleCraAsCbsStattFlag y NoIncorrectPicOutputFlag se ajustan a cero. Las operaciones anteriores, tanto para imágenes de IRAP como para imágenes de GDR, se utilizan para la identificación de los CVS en el flujo de bits. La decodificación se invoca repetidamente para cada imagen codificada en BitstreamToDecode en orden de decodificación.

Un proceso de decodificación de ejemplo para la construcción de listas de imágenes de referencia es el siguiente. Este proceso se invoca al comienzo del proceso de decodificación para cada segmento de una imagen no de IDR. Las imágenes de referencia se abordan a través de índices de referencia. Un índice de referencia es un índice de una lista de imágenes de referencia. Cuando se decodifica un segmento I, no se utiliza ninguna lista de imágenes de referencia para decodificar los datos del segmento. Cuando se decodifica un segmento P, solo se utiliza la lista de imágenes de referencia 0 (por ejemplo, RefPicList[0]) para decodificar los datos del segmento. Cuando se decodifica un segmento B, tanto la lista de imágenes de referencia 0 como la lista de imágenes de referencia 1 (por ejemplo, RefPicList[1]) se utilizan para decodificar los datos del segmento.

Las siguientes restricciones se aplican a la conformidad del flujo de bits. Para cada i igual a cero o uno, num_ref_entries[i][RplsIdx[i]] no debe ser menor que NumRefIdxActive[i]. La imagen a la que hace referencia cada entrada activa en RefPicList[0] o RefPicList[1] debe estar presente en la DPB y debe tener un TemporalId menor o igual que el de la imagen actual. La imagen a la que hace referencia cada entrada en RefPicList[0] o RefPicList[1] no debe ser la imagen actual y debe tener non_reference_picture_flag igual a cero. Una entrada de imagen de referencia a corto plazo (Short Term Reference Picture, STRP) en RefPicList[0] o RefPicList[1] de un segmento de una imagen y una entrada de imagen de referencia a largo plazo (LTRP) en RefPicList[0] o RefPicList[1] del mismo segmento o de un segmento diferente de la misma imagen no debe hacer referencia a la misma imagen. No debe haber ninguna entrada de LTRP en RefPicList[0] o RefPicList[1] para la cual la diferencia entre PicOrderCntVal de la imagen actual y PicOrderCntVal de la imagen a la que hace referencia la entrada sea mayor o igual a doscientos veinticuatro.

Sea setOfRefPics el conjunto de imágenes únicas a las que hacen referencia todas las entradas en RefPicList[0] que tienen el mismo nuh_layer_id que la imagen actual y todas las entradas en RefPicList[1] que tienen el mismo nuh_layer_id que la imagen actual. El número de imágenes en setOfRefPics debe ser menor o igual que MaxDecPicBuffMinus1 y setOfRefPics debe ser el mismo para todos los segmentos de una imagen. Cuando la imagen actual es una imagen de acceso a subcapa temporal por etapas (Step-wise Temporal Sublayer Access, STSA), no debe haber ninguna entrada activa en RefPicList[0] o RefPicList[1] que tenga TemporalId igual al de la imagen actual. Cuando la imagen actual es una imagen que sigue, en orden de decodificación, a una imagen de STSA que tiene TemporalId igual al de la imagen actual, no habrá ninguna imagen que tenga TemporalId igual al de la imagen actual incluida como entrada activa en RefPicList [0] o RefPicList[1] que precede a la imagen de STSA en orden de decodificación.

La imagen a la que hace referencia cada entrada de imagen de referencia entre capas (ILRP) en RefPicList[0] o RefPicList[1] de un segmento de la imagen actual estará en la misma unidad de acceso que la imagen actual. La imagen a la que hace referencia cada entrada de ILRP en RefPicList[0] o RefPicList[1] de un

segmento de la imagen actual estará presente en la DPB y tendrá un nuh_layer_id menor que el de la imagen actual. Cada entrada de ILRP en RefPicList[0] o RefPicList[1] de un segmento debe ser una entrada activa.

Un ejemplo de especificación de HRD es el siguiente. El HRD se utiliza para comprobar la conformidad del flujo de bits y del decodificador. Se utiliza un conjunto de pruebas de conformidad de flujo de bits para verificar la conformidad de un flujo de bits, al que se hace referencia como flujo de bits completo, denominado flujo de bits completo. El conjunto de pruebas de conformidad de flujo de bits sirve para probar la conformidad de cada OP de cada OLS especificado por el VPS.

Para cada prueba, se aplican las siguientes etapas ordenadas en el orden indicado, seguidas de los procesos descritos después de estas etapas en este apartado. Un punto de operación bajo prueba, denominado targetOp, se selecciona seleccionando un OLS objetivo con un índice de OLS opOlsIdx y un valor de TemporalId más alto, opTid. El valor de opOlsIdx está en el rango de cero a TotalNumOlss - 1, inclusive. El valor de opTid está en el rango de cero a vps_max_sub capas menos1, inclusive. Cada par de valores seleccionados de opOlsIdx y opTid será tal que el subflujo de bits que es la salida al invocar el proceso de extracción del subflujo de bits con EntireBitstream, opOlsIdx y opTid como entradas satisfaga las siguientes condiciones. Hay al menos una unidad de NAL de VCL con nuh_layer_id igual a cada uno de los valores de nuh_layer_id en LayerIdInOls[opOlsIdx] en BitstreamToDecode. Hay al menos una unidad de NAL de VCL con TemporalId igual a opTid en BitstreamToDecode.

Si las capas en targetOp incluyen todas las capas en entireBitstream y opTid es igual o mayor que el valor de TemporalId más alto entre todas las unidades de NAL en entireBitstream, BitstreamToDecode se configura para que sea idéntico a entireBitstream. De lo contrario, BitstreamToDecode se configura como salida invocando el proceso de extracción de subflujo de bits con entireBitstream, opOlsIdx y opTid como entradas. Los valores de TargetOlsIdx y Htid se ajustan iguales a opOlsIdx y opTid, respectivamente, de targetOp. Se selecciona un valor de ScIdx. El ScIdx seleccionado estará en el rango de cero a hrd_cpb_cnt_minus1, inclusive. Una unidad de acceso en BitstreamToDecode asociada con mensajes de SEI del período de almacenamiento en memoria intermedia (presentes en TargetLayerBitstream o disponibles a través de mecanismos externos) aplicables a TargetOlsIdx se selecciona como el punto de inicialización de HRD y se denomina unidad de acceso cero para cada capa en el OLS de destino.

Las etapas posteriores se aplican a cada capa con el índice de capa de OLS TargetOlsLayerIdx en el OLS de destino. La estructura de sintaxis ols_hrd_parameters( ) y la estructura de sintaxis sublayer_hrd_parameters( ) aplicables a BitstreamToDecode se seleccionan de la siguiente manera. Se selecciona la estructura de sintaxis ols_hrd_parameters( ) de orden ols_hrd_idx[TargetOlsIdx]en el VPS (o proporcionada a través de un mecanismo externo). Dentro de la estructura de sintaxis ols_hrd_parameters( ) seleccionada, si BitstreamToDecode es un flujo de bits de Tipo I, se selecciona la estructura de sintaxis sub_layer_hrd_parameters(Htid) que sigue inmediatamente a la condición si se selecciona (general_vcl_hrd_params_present_flag) y la variable NalHrdModeFlag se ajusta a cero. De lo contrario (BitstreamToDecode es un flujo de bits de Tipo II), la estructura de sintaxis sub_layer_hrd_parameters(Htid) que sigue inmediatamente se selecciona la condición si (general_vcl_ hrd_params_present_flag) (en este caso la variable NalHrdModeFlag se ajusta a cero) o la condición si (general_nal_hrd_params_present_flag) (en este caso la variable NalHrdModeFlag se ajusta a 1). Cuando BitstreamToDecode es un flujo de bits de Tipo II y NalHrdModeFlag es igual a cero, todas las unidades de NAL no de VCL, excepto las unidades de NAL de datos de relleno, y todos los elementos de sintaxis leading_zero_8bits, zero_byte, start_code_prefix_one_3bytes y trailing_zero_8bits que forman un flujo de bytes a partir del flujo de unidades de NAL, cuando están presentes, se descartan de BitstreamToDecode y el flujo de bits restante se asigna a BitstreamToDecode.

Cuando el indicador hrd_params_present de la unidad de decodificación es igual a uno, la CPB está programada para funcionar ya sea al nivel de la unidad de acceso (en cuyo caso la variable DecodingUnitHrdFlag se ajusta a cero) o al nivel de la unidad de decodificación (en cuyo caso la variable DecodingUnitHrdFlag se ajusta a uno). De lo contrario, DecodingUnitHrdFlag se ajusta a cero y la CPB está programada para funcionar al nivel de unidad de acceso.

Para cada unidad de acceso en BitstreamToDecode a partir de la unidad de acceso cero, se selecciona el mensaje de SEI del período de almacenamiento en memoria intermedia (presente en BitstreamToDecode o disponible a través de mecanismos externos) que está asociado con la unidad de acceso y se aplica a TargetOlsIdx, se selecciona el mensaje de SEI de temporización de imagen (presente en BitstreamToDecode o disponible a través de mecanismos externos) que está asociado con la unidad de acceso y se aplica a TargetOlsIdx, y cuando DecodingUnitHrdFlag es igual a uno y Decoding_unit_cpb_params_in_pic_timing_sei_flag es igual a cero, se seleccionan los mensajes de SEI de información de la unidad de decodificación de (presentes en BitstreamToDecode o disponibles a través de mecanismos externos) que están asociados con unidades de decodificación en la unidad de acceso y se aplican a TargetOlsIdx.

Cada prueba de conformidad incluye una combinación de una opción en cada una de las etapas anteriores. Cuando hay más de una opción para una etapa, para cualquier prueba de conformidad particular solo se elige una opción. Todas las combinaciones posibles de todas las etapas forman el conjunto completo de pruebas de conformidad. Para cada punto de operación bajo prueba, el número de pruebas de conformidad del flujo de bits que se realizarán es igual a n0 * n1 * n2 * n3, donde los valores de n0, n1, n2 y n3 se especifican de la siguiente manera: n1 es igual a hrd_cpb_cnt_minus1 + 1; n1 es el número de unidades de acceso en BitstreamToDecode que están asociadas con mensajes de SEI del período de almacenamiento en memoria intermedia y n2 se obtiene de la siguiente manera. Si BitstreamToDecode es un flujo de bits de Tipo I, n0 es igual a uno. De lo contrario (BitstreamToDecode es un flujo de bits de Tipo II), n0 es igual a dos. n3 se obtiene de la siguiente manera. Si decoding_unit_hrd_params_present_flag es igual a cero, n3 es igual a uno. En caso contrario, n3 es igual a dos.

El HRD contiene un extractor de flujo de bits (opcionalmente presente), una memoria intermedia de imágenes codificadas (CPB), un proceso de decodificación instantánea, una memoria intermedia de imágenes decodificadas (Decoded Picture Buffer, DPB) que conceptualmente contiene una sub-DPB para cada capa, y recorte de salida. Para cada prueba de conformidad de flujo de bits, el tamaño de CPB (número de bits) es CpbSize[Htid][ScIdx], y los parámetros de DPB max_dec_pic_buffering_minus1[Htid], max_num_reorder_pics[Htid] y MaxLatencyPictures[Htid] para cada capa se encuentran o se obtienen a partir de la estructura de sintaxis dpb_parameters( ) que se aplica a la capa dependiendo de si la capa es una capa independiente y si la capa es una capa de salida del OLS de destino.

El HRD puede funcionar de la siguiente manera. El HRD se inicializa en la unidad de decodificación cero, estando tanto la CPB como cada sub-DPB de la DPB configuradas para que estén vacías (el estado de llenado de la sub-DPB para cada sub-DPB se ajusta igual a cero). Después de la inicialización, el HRD no puede ser reinicializado mediante mensajes de SEI posteriores al período de almacenamiento en memoria intermedia. Los datos asociados con las unidades de decodificación que fluyen hacia cada CPB de acuerdo con un cronograma de llegada específico son entregados por el programador de flujo hipotético (HSS). Los datos asociados con cada unidad de decodificación se eliminan y decodifican instantáneamente mediante el proceso de decodificación instantánea en el momento de eliminación de la CPB de la unidad de decodificación. Cada imagen decodificada se coloca en la DPB. Una imagen decodificada se elimina de la DPB cuando la imagen decodificada ya no es necesaria para la referencia de interpredicción y ya no es necesaria para la salida.

Una operación de ejemplo de la memoria intermedia de imágenes decodificadas es la siguiente. Estas especificaciones pueden aplicarse independientemente a cada conjunto de parámetros de la memoria intermedia de imágenes decodificadas (DPB) seleccionado. La memoria intermedia de imágenes decodificadas incluye conceptualmente sub-DPB, y cada sub-DPB contiene memorias intermedias de almacenamiento de imágenes para el almacenamiento de imágenes decodificadas de una capa. Cada una de las memorias intermedias de almacenamiento de imágenes puede contener una imagen decodificada que se marca como utilizada como referencia o se conserva para una salida posterior. Los procesos descritos en la presente memoria se aplican secuencialmente y se aplican de manera independiente para cada capa, comenzando desde la capa más baja en el OLS, en orden creciente de valores de nuh_layer_id de las capas en el OLS. Cuando estos procesos se aplican para una capa particular, solo se ve afectada la sub-DPB de esa capa particular. En las descripciones de estos procesos, la DPB se refiere a la sub-DPB para la capa particular, y la capa particular se denomina capa actual.

En la operación del DPB de temporización de salida, las imágenes decodificadas con PicOutputFlag igual a uno en la misma unidad de acceso se generan consecutivamente en orden ascendente de los valores de nuh_layer_id de las imágenes decodificadas. Sea la imagen n y la imagen actual la imagen codificada o la imagen decodificada de la unidad de acceso n para un valor particular de nuh_layer_id, en donde n es un número entero no negativo. La eliminación de imágenes de la DPB antes de decodificar la imagen actual se produce de la siguiente manera. La eliminación de imágenes de la DPB antes de la decodificación de la imagen actual (pero después de analizar sintácticamente la cabecera del primer segmento de la imagen actual) ocurre sustancialmente de manera instantánea en el momento de la eliminación de la CPB de la primera unidad de decodificación de la unidad de acceso n (que contiene la imagen actual) y continúa de la siguiente manera.

Se invoca el proceso de decodificación para la construcción de la lista de imágenes de referencia, y se invoca el proceso de decodificación para el marcado de imágenes de referencia. Cuando la AU actual es una AU de inicio de secuencia de vídeo codificada (CVSS) que no es AU cero, se aplican las siguientes etapas ordenadas. La variable NoOutputOfPriorPicsFlag se obtiene para el decodificador bajo prueba de la siguiente manera. Si el valor de pic_width_max_in_luma_samples, pic_height_max_in_luma_samples, chroma_format_idc, separate_colour_plane_flag, bit_depth_luma_minus8, bit_depth_chroma_minus8 o max_dec_pic_buffering_minus1[Htid] obtenido para cualquier imagen en la AU actual es diferente del valor de pic_width_in_luma_samples, pic_height_in_luma_samples, chroma format idc, separate_colour_plane_flag, bit_depth_luma_minus8, bit_depth_chroma_minus8 o max_dec_pic_buffering_minus1[Htid], respectivamente, obtenidos para la imagen anterior en el mismo CLVS, el decodificador sometido a prueba puede ajustar NoOutputOfPriorPicsFlag a uno, independientemente del valor de no_output_of_prior_pics_flag. Aunque en

estas condiciones puede ser preferible ajustar NoOutputOfPriorPicsFlag igual a no_output_of_prior_pics_flag, en este caso el decodificador bajo prueba puede ajustar NoOutputOfPriorPicsFlag a uno. De lo contrario, NoOutputOfPriorPicsFlag se ajusta a no_output_of_prior_pics_flag.

El valor de NoOutputOfPriorPicsFlag obtenido para el decodificador bajo prueba se aplica para el HRD, de tal manera que cuando el valor de NoOutputOfPriorPicsFlag es igual a uno, todas las memorias intermedias de almacenamiento de imágenes en la DPB se vacían sin generar las imágenes que contienen, y el estado de llenado de la DPB se ajusta a cero. Cuando se cumplen las dos condiciones siguientes para cualquier imagen k en la DPB, todas esas imágenes k en la DPB se eliminan de la DPB. La imagen k puede marcarse como no utilizada como referencia, o la imagen k puede tener un PictureOutputFlag igual a cero o un tiempo de salida de DPB es menor o igual al tiempo de eliminación de la CPB de la primera unidad de decodificación (indicada como unidad de decodificación m) de la actual imagen n, donde DpbOutputTime[k] es menor o igual que DuCpbRemovalTime[m]. Por cada imagen que se elimina de la DPB, el estado de llenado de la DPB se reduce en uno.

El funcionamiento de la orden de salida de DPB puede ser el siguiente. Estos procesos pueden aplicarse independientemente a cada conjunto de parámetros de la memoria intermedia de imágenes decodificadas (DPB) seleccionados. La memoria intermedia de imágenes decodificadas incluye conceptualmente varias sub-DPB, y cada sub-DPB contiene memorias intermedias de almacenamiento de imágenes para el almacenamiento de imágenes decodificadas de una capa. Cada una de las memorias intermedias de almacenamiento de imágenes puede contener una imagen decodificada que se marca como utilizada como referencia o se conserva para una salida posterior. Se invoca el proceso para la salida y eliminación de imágenes de la DPB antes de decodificar la imagen actual, seguido por la invocación del proceso para marcar y almacenar la imagen decodificada actual, y finalmente seguido por la invocación del proceso para cambios adicionales. Estos procesos se aplican de manera independiente para cada capa, comenzando desde la capa más baja en el OLS, en orden creciente de los valores de nuh_layer_id de las capas en el OLS. Cuando estos procesos se aplican para una capa particular, solo se ve afectada la sub-DPB de esa capa particular.

En la operación de la DPB de temporización de salida, las imágenes decodificadas con PicOutputFlag igual a uno en la misma unidad de acceso se generan consecutivamente en orden ascendente de los valores de nuh_layer_id de las imágenes decodificadas. Sea la imagen n y la imagen actual la imagen codificada o la imagen decodificada de la unidad de acceso n para un valor particular de nuh_layer_id, en donde n es un número entero no negativo. La salida y eliminación de imágenes de la DPB se describe a continuación.

La generación y eliminación de imágenes de la DPB antes de la decodificación de la imagen actual (pero después de analizar la cabecera del primer segmento de la imagen actual) ocurre sustancialmente de manera instantánea cuando la primera unidad de decodificación de la unidad de acceso que contiene la imagen actual se elimina de la CPB, y continúa de la siguiente manera. Se invoca el proceso de decodificación para la construcción de la lista de imágenes de referencia, y se invoca el proceso de decodificación para el marcado de imágenes de referencia. Si la AU actual es una AU de CVSS que no es AU cero, se aplican las siguientes etapas ordenadas. La variable NoOutputOfPriorPicsFlag se obtiene para el decodificador bajo prueba de la siguiente manera. Si el valor de pic_width_max_in_luma_samples, pic_height_max_in_luma_samples, chroma_format_idc, separate_colour_plane_flag, bit_depth_luma_minus8, bit_depth_chroma_minus8 o max_dec_pic_buffering_minus1[Htid] obtenido para cualquier imagen en la AU actual es diferente del valor de pic_width_in_luma_samples, pic_height_in_luma_samples, chroma format idc, separate colour_plane_flag, bit_depth_luma_minus8, bit_depth_chroma_minus8 o max_dec_pic_buffering_minus1[Htid], respectivamente, obtenidos para la imagen anterior en el mismo CLVS, el decodificador sometido a prueba puede ajustar NoOutputOfPriorPicsFlag a uno, independientemente del valor de no_output_of_prior_pics_flag.

Aunque en estas condiciones puede ser preferible ajustar NoOutputOfPriorPicsFlag igual a no_output_of_prior_pics_flag, en este caso el decodificador bajo prueba puede ajustar NoOutputOfPriorPicsFlag a uno. De lo contrario, NoOutputOfPriorPicsFlag se ajusta a no_output_of_prior_pics_flag. El valor de NoOutputOfPriorPicsFlag obtenido para el decodificador sometido a prueba se aplica al HRD de la siguiente manera. Si NoOutputOfPriorPicsFlag es igual a uno, todas las memorias intermedias de almacenamiento de imágenes en la DPB se vacían sin salida de las imágenes que contienen y el estado de llenado de la DPB se ajusta a cero. De lo contrario (NoOutputOfPriorPicsFlag es igual a cero), todas las memorias intermedias de almacenamiento de imágenes que contienen una imagen marcada como no necesaria para la salida y no utilizada como referencia, se vacían (sin salida), y todas las memorias intermedias de almacenamiento de imágenes que no están vacías en la DPB se vacían invocando repetidamente un cambio en el contador y el estado de llenado de la DPB se ajusta a cero.

De lo contrario (la imagen actual no es una imagen de CLVSS), todas las memorias intermedias de almacenamiento de imágenes que contienen una imagen que están marcadas como no necesarias para la salida y no utilizadas como referencia, se vacían (sin salida). Por cada memoria intermedia de almacenamiento de imágenes que se vacía, el estado de llenado de la DPB se reduce en uno. Cuando una o más de las siguientes condiciones son verdaderas, el proceso de cambio en el contador se invoca repetidamente mientras

se disminuye aún más el estado de llenado de la DPB en uno por cada memoria intermedia de almacenamiento de imágenes adicional que se vacía, hasta que ninguna de las siguientes condiciones sea verdadera. La cantidad de imágenes en la DPB que están marcadas como de salida necesaria es mayor que max_num_reorder_pics[Htid]. max_latency_increase_plus1[Htid] no es igual a cero y hay al menos una imagen en la DPB que está marcada como de salida necesaria, para la cual la variable asociada PicLatencyCount es mayor o igual que MaxLatencyPictures[Htid]. El número de imágenes en la DPB es mayor o igual que max_dec_pic_buffering_minus1[Htid] + 1.

En un ejemplo, pueden producirse cambios en el contador adicionales de la siguiente manera. Los procesos especificados pueden ocurrir sustancialmente de manera instantánea cuando la última unidad de decodificación de la unidad de acceso n que contiene la imagen actual se elimina de la CPB. Cuando la imagen actual tiene PictureOutputFlag igual a uno, para cada imagen en la DPB que está marcada como de salida necesaria y sigue a la imagen actual en el orden de salida, la variable asociada PicLatencyCount se ajusta a PicLatencyCount + 1. También se aplica lo siguiente. Si la imagen decodificada actual tiene PictureOutputFlag igual a uno, la imagen decodificada actual se marca como de salida necesaria y una variable asociada PicLatencyCount se ajusta a cero. De lo contrario (la imagen decodificada actual tiene PictureOutputFlag igual a cero), la imagen decodificada actual se marca como de salida no necesaria.

Cuando una o más de las siguientes condiciones son verdaderas, el proceso de cambio en el contador se invoca repetidamente hasta que ninguna de las siguientes condiciones sea verdadera. La cantidad de imágenes en la DPB que están marcadas como de salida necesaria es mayor que max_num_reorder_pics[Htid]. max_latency_increase_plus1[Htid] no es igual a cero y hay al menos una imagen en la DPB que está marcada como de salida necesaria, para la cual la variable asociada PicLatencyCount es mayor o igual que MaxLatencyPictures[Htid].

El proceso de cambio en el contador incluye las siguientes etapas ordenadas. La imagen o imágenes que se envían primero se seleccionan como las que tienen el valor más pequeño de PicOrderCntVal de all_pictures en la DPB marcado como de salida necesaria. Cada una de estas imágenes, en orden ascendente de nuh_layer_id, se recorta, utilizando la ventana de recorte de conformidad para la imagen, la imagen recortada se genera y la imagen se marca como de salida no necesaria. Cada memoria intermedia de almacenamiento de imágenes que contiene una imagen marcada como no utilizada como referencia y que fue una de las imágenes recortadas y producidas se vacía, y el estado de llenado de la sub-DPB asociada se reduce en uno. Para dos imágenes cualesquiera, picA y picB, que pertenecen al mismo CVS y se generan mediante el proceso de cambio en el contador, cuando picA se genera antes que picB, el valor de PicOrderCntVal de picA es menor que el valor de PicOrderCntVal de picB.

Un ejemplo de proceso de extracción de subflujo de bits es el siguiente. Las entradas a este proceso son un flujo de bits inBitstream, un índice de OLS objetivo targetOlsIdx y un valor más alto de TemporalId de destino tIdTarget. La salida de este proceso es un subflujo de bits outBitstream. La conformidad del flujo de bits puede requerir que, para cualquier flujo de bits de entrada, un subflujo de bits de salida que sea la salida de este proceso con el flujo de bits, targetOlsIdx sea igual a un índice de la lista de OLS especificada por el VPS, y tIdTarget sea igual a cualquier valor en el rango de cero a seis, inclusive, como entradas, y que satisfaga las siguientes condiciones será un flujo de bits conforme. El subflujo de bits de salida contiene al menos una unidad de NAL de VCL con una nuh_layer_id igual a cada uno de los valores de nuh_layer_id en LayerIdInOls[targetOlsIdx]. El subflujo de bits de salida contiene al menos una unidad de NAL de VCL con TemporalId igual a tIdTarget. Un flujo de bits conforme contiene una o más unidades de NAL de segmento codificadas con TemporalId igual a cero, pero no tiene que contener unidades de NAL de segmento codificadas con nuh_layer_id igual a cero.

El subflujo de bits de salida Outbitstream se obtiene de la siguiente manera. El flujo de bits outBitstream está configurado para que sea idéntico al flujo de bits inBitstream. Todas las unidades de NAL con TemporalId mayor que tIdTarget se eliminan de outBitstream. Todas las unidades de NAL con un nuh_layer_id no incluido en la lista, LayerIdInOls[targetOlsIdx] se eliminan de outBitstream. Todas las unidades de NAL de SEI que contienen un mensaje de SEI anidado escalable que tiene nesting_ols_flag igual a uno y no hay ningún valor de i en el rango de cero a nesting_num_olss_minus1, inclusive, de tal manera que NestingOlsIdx[i] es igual a targetOlsIdx se eliminan de outBitstream. Cuando targetOlsIdx es mayor que cero, todas las unidades de NAL de SEI que contienen un mensaje de SEI anidado no escalable con payloadType igual a cero (período de almacenamiento en memoria intermedia), uno (temporización de la imagen) o ciento treinta (información de la unidad de decodificación), se eliminan de outBitstream.

Un ejemplo de sintaxis de mensaje de SEI de anidamiento escalable es el siguiente.

scalable_nesting (payloadSize) {	Descriptor
nesting_ols_flag	u(1)

si (nesting_ols_flag) {	
nesting_num_olss_minus1	ue(v)
para (i = 0; i <= nesting_num_olss_minus1; i++)	
nesting_ols_idx_delta_minus1[i]	ue(v)
} si no {	
nesting_all_layers_flag	u(1)
si (!nesting_all_layers_flag) {	
nesting_num_layers_minus1	ue(v)
para (i = 1; i <= nesting_num_layers_minus1; i++)	
nesting_layer_id[i]	u(6)
}	
}	
nesting_num_seis_minus1	ue(v)
mientras(!byte_aligned( ))	
nesting_zero_bit_/* igual a 0 */	u(1)
para (i = 0; i <= nesting_num_seis_minus1; i++)	
sei_message( )	
}	

Un ejemplo de semántica de carga útil de SEI general es el siguiente. Lo siguiente se aplica a las capas aplicables u al OLS de mensajes de SEI anidados no escalables. Para un mensaje de SEI anidado no escalable, cuando payloadType es igual a cero (período de almacenamiento en memoria intermedia), uno (temporización de imagen) o ciento treinta (información de la unidad de decodificación), el mensaje de SEI anidado no escalable se aplica solo al OLS de orden 0. Para un mensaje de SEI anidado no escalable, cuando payloadType es igual a cualquier valor entre VclAssociatedSeiList, el mensaje de SEI anidado no escalable se aplica solo a la capa para la cual las unidades de NAL de VCL tienen nuh_layer_id igual al nuh_layer_id de la unidad de NAL de SEI que contiene el mensaje de SEI.

La conformidad del flujo de bits puede requerir que se apliquen las siguientes restricciones al valor de nuh_layer_id de las unidades de NAL de SEI. Cuando un mensaje de SEI anidado no escalable tiene un tipo de carga útil igual a cero (período de almacenamiento en memoria intermedia), uno (temporización de imagen) o ciento treinta (información de la unidad de decodificación), la unidad de NAL de SEI que contiene el mensaje de SEI anidado no escalable debe tener un nuh_layer_id igual a vps_layer_id[0]. Cuando un mensaje de SEI anidado no escalable tiene payloadType igual a cualquier valor entre VclAssociatedSeiList, la unidad de NAL de SEI que contiene el mensaje de SEI anidado no escalable debe tener nuh_layer_id igual al valor de nuh_layer_id de la unidad de NAL de VCL asociada con la unidad de NAL de SEI. Una unidad de NAL de SEI que contiene un mensaje de SEI anidado escalable debe tener un nuh_layer_id igual al valor más bajo del nuh_layer_id de todas las capas a las que se aplica el mensaje de SEI anidado escalable (cuando nesting_ols_flag del mensaje de SEI anidado escalable es igual a cero) o el valor más bajo de nuh_layer_id de todas las capas en los OLS a los que se aplica el mensaje de SEI anidado escalable (cuando nesting_ols_flag del mensaje de SEI anidado escalable es igual a uno).

Un ejemplo de semántica de mensaje de SEI anidado escalable es el siguiente. El mensaje de SEI anidado escalable proporciona un mecanismo para asociar mensajes de SEI con OLS específicos o con capas específicas. Un mensaje de SEI anidado escalable contiene uno o más mensajes de SEI. Los mensajes de SEI contenidos en el mensaje de SEI anidado escalable también se denominan mensajes de SEI anidados escalables. La conformidad con el flujo de bits puede requerir que se apliquen las siguientes restricciones al contenido de mensajes de SEI en un mensaje de SEI anidado escalable.

Un mensaje de SEI que tiene un tipo de carga útil igual a ciento treinta y dos (hash de imagen decodificada) o ciento treinta y tres (anidamiento escalable) puede no estar contenido en un mensaje de SEI de anidamiento escalable. Cuando un mensaje de SEI anidado escalable contiene un mensaje de SEI de período de almacenamiento en memoria intermedia, temporización de imagen o información de unidad de decodificación, el mensaje de SEI de anidamiento escalable no debe contener ningún otro mensaje de SEI con tipo de carga útil no igual a cero (período de almacenamiento en memoria intermedia), uno (temporización de imagen), o ciento treinta (información de la unidad de decodificación).

La conformidad del flujo de bits puede requerir que se apliquen las siguientes restricciones al valor de

nal_unit_type de la unidad de NAL de SEI que contiene un mensaje de SEI anidado escalable. Cuando un mensaje de SEI anidado escalable contiene un mensaje de SEI que tiene un tipo de carga útil igual a cero (período de almacenamiento en memoria intermedia), uno (temporización de la imagen), ciento treinta (información de la unidad de decodificación), ciento cuarenta y cinco (indicación de RAP dependiente) o ciento sesenta y ocho (información del campo de trama), la unidad de NAL de SEI que contiene el mensaje de SEI anidado escalable debe tener un nal_unit_type igual a PREFIX_SEI_NUT.

Un nesting_ols_flag igual a uno especifica que los mensajes de SEI anidados escalables se aplican a OLS específicos. Un nesting_ols_flag igual a cero especifica que los mensajes de SEI anidados escalables se aplican a capas específicas. La conformidad del flujo de bits puede requerir que se apliquen las siguientes restricciones al valor de nesting_ols_flag. Cuando el mensaje de SEI de anidamiento escalable contiene un mensaje de SEI que tiene un tipo de carga útil igual a cero (período de almacenamiento en memoria intermedia), uno (temporización de la imagen) o ciento treinta (información de la unidad de decodificación), el valor de nesting_ols_flag debe ser igual a uno. Cuando el mensaje de SEI de anidamiento escalable contiene un mensaje de SEI que tiene payloadType igual a un valor en VclAssociatedSeiList, el valor de nesting_ols_flag debe ser igual a cero. El nesting_num_olss_minus1 + 1 especifica el número de OLS a los que se aplican los mensajes de SEI anidados escalables. El valor de nesting_num_olss_minus1 debe estar en el rango de 0 a TotalNumOlss - 1, inclusive.

Se utiliza un nesting_ols_idx_delta_minus1[i] para obtener la variable NestingOlsIdx[i] que especifica el índice de OLS del OLS de orden i al que se aplican los mensajes de SEI anidados escalables cuando nesting_ols_flag es igual a uno. El valor de nesting_ols_idx_delta_minus1[i] debe estar en el rango de cero a TotalNumOlss menos dos, inclusive. La variable NestingOlsIdx[i] se puede obtener de la siguiente manera.

```
si( i == 0 )
        NestingOlsIdx[ i ] = nesting_ols_idx_delta_minus1[ i ]
si no
        NestingOlsIdx[ i ] = NestingOlsIdx[ i − 1 ] + nesting_ols_idx_delta_minus1[ i ] + 1
```

Un nesting_all_layers_flag igual a uno especifica que los mensajes de SEI anidados escalables se aplican a todas las capas que tienen un nuh_layer_id mayor o igual que el nuh_layer_id de la unidad de NAL de SEI actual. Un indicador de anidación de todas las capas igual a cero especifica que los mensajes de SEI anidados escalables pueden o no aplicarse a todas las capas que tienen un nuh_layer_id mayor o igual que el nuh_layer_id de la unidad de NAL de SEI actual. Un nesting_num_olss_minus1 + 1 especifica el número de capas a las que se aplican los mensajes de SEI anidados escalables. El valor de nesting_num_layers_minus1 debe estar en el rango de cero a vps_max_layers_minus1 - GeneralLayerIdx[nuh_layer_id], inclusive, donde nuh_layer_id es el nuh_layer_id de la unidad de NAL de SEI actual. Un nesting_layer_id[i] especifica el valor de nuh_layer_id de la capa de orden i a la que se aplican los mensajes de SEI anidados escalables cuando nesting_all_layers_flag es igual a cero. El valor de nesting_layer_id[i] debe ser mayor que el nuh_layer_id, donde el nuh_layer_id es el nuh_layer_id de la unidad de NAL de SEI actual.

Cuando nesting_ols_flag es igual a cero, la variable NestingNumLayers, que especifica el número de capas a las que se aplican los mensajes de SEI anidados escalables, y la lista NestingLayerId[i] para i en el rango de cero a NestingNumLayers - 1, inclusive, especificando la lista de valores nuh_layer_id de las capas a las que se aplican los mensajes de SEI anidados escalables, se puede obtener de la siguiente manera, donde nuh_layer_id es el nuh_layer_id de la unidad de NAL de SEI actual si (nesting_all_layers_flag) {

```
        NestingNumLayers = vps_max_layers_minus1 + 1 − GeneralLayerIdx[ nuh_layer_id ]
    para ( i = 0; i < NestingNumLayers; i ++)
            NestingLayerId[ i ] = vps_layer_id[ GeneralLayerIdx[ nuh_layer_id ] + i ]
} si no {
        NestingNumLayers = nesting_num_layers_minus1 + 1
    para ( i = 0; i < NestingNumLayers; i ++)
            NestingLayerId[ i ] = ( i == 0 ) ? nuh_layer_id : nesting_layer_id[ i ]
}
```

Un nesting_num_seis_minus1 + 1 especifica el número de mensajes de SEI anidados escalables. El valor de

nesting_num_seis_minus1 debe estar en el rango de cero a sesenta y tres inclusive. El nesting_zero_bit debe ser igual a cero.

La figura 8 es un diagrama esquemático de un dispositivo de codificación de vídeo 800 de ejemplo. El dispositivo de codificación de vídeo 800 es adecuado para implementar los ejemplos/realizaciones divulgados tal como se describen en la presente memoria. El dispositivo de codificación de vídeo 800 comprende puertos de flujo descendente 820, puertos de flujo ascendente 850 y/o unidades transceptoras (Tx/Rx) 810, incluidos transmisores y/o receptores para comunicar datos de flujo ascendente y/o de flujo descendente a través de una red. El dispositivo de codificación de vídeo 800 también incluye un procesador 830 que incluye una unidad lógica y/o unidad central de procesamiento (CPU) para procesar los datos, y una memoria 832 para almacenar los datos. El dispositivo de codificación de vídeo 800 también puede comprender componentes eléctricos, ópticos a eléctricos (OE), eléctricos a ópticos (EO), y/o componentes de comunicación inalámbrica acoplados a los puertos de flujo ascendente 850 y/o a los puertos de flujo descendente 820 para comunicación de datos a través de redes de comunicación eléctricas, ópticas o inalámbricas. El dispositivo de codificación de vídeo 800 también puede incluir dispositivos de entrada y/o salida (E/S) 860 para comunicar datos hacia y desde un usuario. Los dispositivos de E/S 860 pueden incluir dispositivos de salida, tales como una pantalla para mostrar datos de vídeo, altavoces para generar datos de audio, etc. Los dispositivos de E/S 860 también pueden incluir dispositivos de entrada, tales como un teclado, ratón, rueda de desplazamiento, etc., y/o interfaces correspondientes para interactuar con tales dispositivos de salida.

El procesador 830 está implementado mediante hardware y software. El procesador 830 puede implementarse como uno o más chips de CPU, núcleos (por ejemplo, como un procesador multinúcleo), una matriz de puertas programable en campo (Field-Programmable Gate Array, FPGA), circuitos integrados de aplicación específica (Application Specific Integrated Circuit, ASIC), y procesadores de señales digitales (Digital Signal Processor, DSP). El procesador 830 está en comunicación con los puertos de flujo descendente 820, Tx/Rx 810, los puertos de flujo ascendente 850 y la memoria 832. El procesador 830 comprende un módulo de codificación 814. El módulo de codificación 814 implementa las realizaciones divulgadas descritas en la presente memoria, tales como los métodos 100, 900 y 1000, que pueden emplear una secuencia de vídeo multicapa 600, un flujo de bits 700 y/o un subflujo de bits 701. El módulo de codificación 814 también puede implementar cualquier otro método/mecanismo descrito en la presente memoria. Además, el módulo de codificación 814 puede implementar un sistema de códec 200, un codificador 300, un decodificador 400 y/o un HRD 500. Por ejemplo, el módulo de codificación 814 puede emplearse para codificar, extraer y/o decodificar un flujo de bits que incluye una capa de transmisión simultánea y ningún VPS. Además, el módulo de codificación 814 puede emplearse para ajustar y/o inferir diversos elementos y/o variables de sintaxis para evitar errores basándose en referencias a un VPS que se extrae como parte de una extracción de subflujo de bits. En consecuencia, el módulo de codificación 814 puede configurarse para realizar mecanismos para abordar uno o más de los problemas explicados anteriormente. Por lo tanto, el módulo de codificación 814 hace que el dispositivo de codificación de vídeo 800 proporcione funcionalidad adicional y/o eficiencia de codificación al codificar datos de vídeo. Por lo tanto, el módulo de codificación 814 mejora la funcionalidad del dispositivo de codificación de vídeo 800 así como aborda problemas que son específicos de las técnicas de codificación de vídeo. Además, el módulo de codificación 814 efectúa una transformación del dispositivo de codificación de vídeo 800 a un estado diferente. Alternativamente, el módulo de codificación 814 puede implementarse como instrucciones almacenadas en la memoria 832 y ejecutadas por el procesador 830 (por ejemplo, como un producto de programa informático almacenado en un medio no transitorio).

La memoria 832 comprende uno o más tipos de memoria tales como discos, unidades de cinta, unidades de estado sólido, memoria de solo lectura (Read Only Memory, ROM), memoria de acceso aleatorio (Random Access Memory, RAM), memoria flash, memoria ternaria direccionable por contenido (Ternary Content-Addressable Memory TCAM), memoria estática de acceso aleatorio (Static RAM, SRAM), etc. La memoria 832 puede usarse como un dispositivo de almacenamiento de datos de desbordamiento, para almacenar programas cuando dichos programas se seleccionan para su ejecución, y para almacenar instrucciones y datos que se leen durante la ejecución del programa.

La figura 9 es un diagrama de flujo de un método de ejemplo 900 de codificación de una secuencia de vídeo multicapa en un flujo de bits, tal como un flujo de bits 700, para soportar la eliminación del VPS 711 durante un proceso de extracción de subflujo de bits 729 para capas de transmisión simultánea. El método 900 puede ser empleado por un codificador, tal como un sistema de códec 200, un codificador 300 y/o un dispositivo de codificación de vídeo 800 al realizar el método 100. Además, el método 900 puede funcionar en un HRD 500 y, por lo tanto, puede realizar pruebas de conformidad en una secuencia de vídeo multicapa 600 y/o una capa extraída de la misma.

El método 900 puede comenzar cuando un codificador recibe una secuencia de vídeo y determina codificar esa secuencia de vídeo en un flujo de bits multicapa, por ejemplo basándose en la entrada del usuario. En la etapa 901, el codificador codifica imágenes codificadas en un conjunto de unidades de NAL de VCL en el flujo de bits. Por ejemplo, el codificador puede codificar las imágenes de la secuencia de vídeo en una o más capas, y codificar las capas en un flujo de bits multicapa. Por tanto, el flujo de bits comprende una o más capas. Una

capa puede incluir un conjunto de unidades de NAL de VCL con el mismo Id de capa y unidades asociadas de NAL no de VCL. Como ejemplo específico, las unidades de NAL de VCL pueden estar asociadas con una capa identificada por/que tiene un nuh_layer_id. Específicamente, un conjunto de unidades de NAL de VCL son parte de una capa cuando todas las unidades de NAL de VCL tienen un valor particular de nuh_layer_id. Una capa puede incluir un conjunto de unidades de NAL de VCL que contienen datos de vídeo de imágenes codificadas, así como cualquier conjunto de parámetros utilizado de codificación de dichas imágenes. Dichos parámetros pueden estar incluidos en un VPS, SPS, PPS, cabecera de imagen, cabecera de segmento u otro conjunto de parámetros o estructura de sintaxis. Como ejemplo específico, el codificador puede codificar en el flujo de bits un SPS que incluye sps_video_parameter_set_id. Una o más de las capas pueden ser capas de salida. Las capas que no son una capa de salida se denominan capas de referencia, y están codificadas para soportar la reconstrucción de la una o varias capas de salida, pero dichas capas de soporte no están destinadas a la salida en un decodificador. De esta manera, el codificador puede codificar diversas combinaciones de capas para su transmisión a un decodificador previa solicitud. La capa se puede transmitir según se desee para permitir que el decodificador obtenga una representación diferente de la secuencia de vídeo dependiendo de las condiciones de la red, las capacidades del hardware y/o la configuración del usuario. En el presente ejemplo, al menos una de las capas es una capa de transmisión simultánea que no utiliza predicción entre capas.

En la etapa 903, un HRD que opera en el codificador puede realizar un conjunto de pruebas de conformidad de flujo de bits en las capas para garantizar la conformidad con VVC u otros estándares. Por ejemplo, el HRD puede obtener un sps_video_parameter_set_id de un SPS. El sps_video_parameter_set_id especifica un valor de vps_video_parameter_set_id de un VPS al que hace referencia el SPS cuando el sps_video_parameter_set_id es mayor que cero. Además, el SPS no hace referencia a un VPS y no se hace referencia a ningún VPS al decodificar cada secuencia de vídeo de capa codificada que hace referencia al SPS cuando el sps_video_parameter_set_id es igual a cero. En consecuencia, el sps_video_parameter_set_id se ajusta a cero y/o se infiere que es cero cuando el sps_video_parameter_set_id se obtiene de un SPS al que hace referencia una secuencia de vídeo de capa codificada contenida en una capa de transmisión simultánea.

El HRD puede ajustar y/o inferir un GeneralLayerIdx[nuh_layer_id] para que sea igual a cero cuando el sps_video_parameter_set_id es igual a cero. El GeneralLayerIdx[nuh_layer_id] es igual a, y por ello indica, un índice de capa actual para una capa correspondiente. Por lo tanto, el índice de capa actual para una capa de transmisión simultánea se ajusta o se infiere que es cero. Además, el HRD puede inferir que un valor de vps_independent_layer_flag[GeneralLayerIdx[nuh_layer_id]] es igual a uno cuando el sps_video_parameter_set_id es igual a cero. Específicamente, un vps_independent_layer_flag [i] está contenido en un VPS y puede ajustarse a cero para indicar que una capa de orden i usa predicción entre capas, o ajustarse a uno para indicar que la capa de orden i no usa predicción entre capas. En consecuencia, vps_independent_layer_flag [GeneralLayerIdx[nuh_layer_id]] especifica si una capa actual con índice GeneralLayerIdx[nuh_layer_id] utiliza predicción entre capas. Cuando la capa actual es una capa de transmisión simultánea, se omite el VPS y no se emplea la predicción entre capas. En consecuencia, la inferencia de un valor de uno cuando el sps_video_parameter_set_id es igual a cero garantiza que la capa de transmisión simultánea funcione correctamente en el HRD y durante la decodificación, evitando al mismo tiempo una referencia al VPS, que se extrae durante la extracción del flujo de bits para las capas de transmisión simultánea. Por lo tanto, la inferencia evita errores de extracción de subflujo de bits que de otro modo ocurrirían cuando se elimina el VPS para una capa de transmisión simultánea. Luego, el HRD puede decodificar la imagen codificada de las unidades de NAL de VCL en la capa de transmisión simultánea basándose en el SPS, el sps_video_parameter_set_id, el GeneralLayerIdx[nuh_layer_id] y/o el vps_independent_layer_flag[GeneralLayerIdx[nuh_layer_id]] para producir una imagen decodificada. Por lo tanto, el HRD puede verificar si la capa de transmisión simultánea del flujo de bits multicapa se ajusta al flujo de bits sin errores inesperados causados por la omisión del VPS para las capas de transmisión simultánea.

En la etapa 905, el codificador puede almacenar el flujo de bits para comunicación hacia un decodificador previa solicitud. El codificador también puede realizar una extracción de subflujo de bits para obtener la capa de transmisión simultánea, y transmitir el flujo de bits/subflujo de bits hacia el codificador según se desee.

La figura 10 es un diagrama de flujo de un método 1000 de ejemplo de decodificación de una secuencia de vídeo de un flujo de bits, tal como un subflujo de bits 701, que incluye una capa de transmisión simultánea extraída de un flujo de bits multicapa, tal como el flujo de bits 700, donde se ha eliminado un VPS 711 durante un proceso de extracción de subflujo de bits 729. El método 1000 puede ser empleado por un decodificador, tal como un sistema de códec 200, un decodificador 400 y/o un dispositivo de codificación de vídeo 800 cuando se realiza el método 100. Además, el método 1000 puede emplearse en una secuencia de vídeo multicapa 600, o una capa extraída de la misma, cuya conformidad ha sido comprobada por un HRD, tal como el HRD 500.

El método 1000 puede comenzar cuando un decodificador comienza a recibir un flujo de bits que contiene una secuencia de vídeo codificada en una capa de transmisión simultánea extraída de un flujo de bits multicapa, por ejemplo como resultado del método 900. En la etapa 1001, el decodificador recibe un flujo de bits que

comprende una capa de transmisión simultánea extraída de un flujo de bits multicapa mediante un codificador u otro servidor de contenido intermedio. La capa de transmisión simultánea contiene una secuencia de vídeo de capa codificada que incluye un conjunto de imágenes codificadas. Por ejemplo, el flujo de bits comprende imágenes codificadas donde cada imagen codificada está incluida en un conjunto de una o más unidades de NAL de VCL asociadas con la capa de transmisión simultánea identificada por/que tiene un nuh_layer_id. Una capa puede incluir un conjunto de unidades de NAL de VCL con el mismo Id de capa y unidades asociadas de NAL no de VCL. Por ejemplo, una capa puede incluir un conjunto de unidades de NAL de VCL que contienen datos de vídeo de imágenes codificadas, así como cualquier conjunto de parámetros utilizado de codificación de dichas imágenes. Por lo tanto, el conjunto de unidades de NAL de VCL es parte de la capa cuando el conjunto de unidades de NAL de VCL tiene un valor particular de nuh_layer_id. La capa de transmisión simultánea también es una capa de salida y no emplea predicción entre capas. El flujo de bits también comprende un SPS que incluye sps_video_parameter_set_id. El sps_video_parameter_set_id especifica un valor de un vps_video_parameter_set_id de un VPS al que hace referencia el SPS cuando el sps_video_parameter_set_id es mayor que cero. Además, el SPS no hace referencia a un VPS y no se hace referencia a ningún VPS al decodificar cada secuencia de vídeo de capa codificada que hace referencia al SPS cuando el sps_video_parameter_set_id es igual a cero. En consecuencia, el sps_video_parameter_set_id se ajusta a cero y/o se infiere que es cero cuando el sps_video_parameter_set_id se obtiene de un SPS al que hace referencia una secuencia de vídeo de capa codificada contenida en una capa de transmisión simultánea. Además, el flujo de bits no contiene un VPS cuando el flujo de bits contiene solo una capa de transmisión simultánea.

En la etapa 1003, el decodificador puede ajustar y/o inferir un GeneralLayerIdx[nuh_layer_id] para que sea igual a cero cuando el sps_video_parameter_set_id es igual a cero. El GeneralLayerIdx[nuh_layer_id] es igual a, y por ello indica, un índice de capa actual para una capa correspondiente. Por lo tanto, el índice de capa actual para una capa de transmisión simultánea se ajusta o se infiere que es cero.

En la etapa 1005, el decodificador puede ajustar/inferir que un valor de vps_independent_layer_flag[GeneralLayerIdx[nuh_layer_id]] es igual a uno cuando el sps_video_parameter_set_id es igual a cero. Específicamente, un vps_independent_layer_flag [i] está contenido en un VPS y puede ajustarse a cero para indicar que una capa de orden i usa predicción entre capas, o ajustarse a uno para indicar que la capa de orden i no usa predicción entre capas. En consecuencia, vps_independent_layer_flag [GeneralLayerIdx[nuh_layer_id]] especifica si una capa actual con índice GeneralLayerIdx[nuh_layer_id] utiliza predicción entre capas. Cuando la capa actual es una capa de transmisión simultánea, se omite el VPS y no se emplea la predicción entre capas. En consecuencia, la inferencia de un valor de uno cuando el sps_video_parameter_set_id es igual a cero garantiza que la capa de transmisión simultánea funcione correctamente durante la decodificación, evitando al mismo tiempo una referencia al VPS, que se extrae durante la extracción del flujo de bits para las capas de transmisión simultánea y, por lo tanto, no se recibe en el decodificador. Por lo tanto, la inferencia evita errores de extracción de subflujo de bits que de otro modo ocurrirían cuando se elimina el VPS para una capa de transmisión simultánea.

En la etapa 1007, el decodificador puede decodificar la imagen codificada de las unidades de NAL de VCL en la capa de transmisión simultánea basándose en el SPS, en el sps_video_parameter_set_id, en el GeneralLayerIdx[nuh_layer_id] y/o en el vps_independent_layer_flag [GeneralLayerIdx[nuh_layer_id]] para producir una imagen decodificada. El decodificador puede entonces reenviar la imagen decodificada para su visualización como parte de una secuencia de vídeo decodificada en la etapa 1009.

La figura 11 es un diagrama esquemático de un sistema 1100 de ejemplo para codificar una secuencia de vídeo multicapa en un flujo de bits 700 para soportar la eliminación del VPS 711 durante el proceso de extracción del subflujo de bits 729 para capas de transmisión simultánea. El sistema 1100 puede implementarse mediante un codificador y un decodificador tal como un sistema de códec 200, un codificador 300, un decodificador 400 y/o un dispositivo de codificación de vídeo 800. Además, el sistema 1100 puede emplear un HRD 500 para realizar pruebas de conformidad en una secuencia de vídeo multicapa 600, un flujo de bits 700 y/o un subflujo de bits 701. Además, el sistema 1100 puede emplearse al implementar el método 100, 900 y/o 1000.

El sistema 1100 incluye un codificador de vídeo 1102. El codificador de vídeo 1102 comprende un módulo de codificación 1103 para la codificación de una imagen codificada y un SPS en un flujo de bits, en donde la imagen codificada está codificada en un conjunto de unidades de NAL de VCL, en donde las unidades de NAL de VCL están asociadas con una capa que tiene un nuh_layer_id, y en donde el SPS incluye un sps_video_parameter_set_id. El codificador de vídeo 1102 comprende además un módulo de HRD 1105 para realizar un conjunto de pruebas de conformidad del flujo de bits en el flujo de bits estableciendo un GeneralLayerIdx[nuh_layer_id] igual a cero cuando el sps_video_parameter_set_id es igual a cero, y decodificando la imagen codificada de las unidades de NAL de VCL basándose en el GeneralLayerIdx[nuh_layer_id] para producir una imagen decodificada. El codificador de vídeo 1102 comprende además un módulo de almacenamiento 1106 para almacenar el flujo de bits para la comunicación hacia un decodificador. El codificador de vídeo 1102 comprende además un módulo de transmisión 1107 para

transmitir el flujo de bits hacia un decodificador de vídeo 1110. El codificador de vídeo 1102 puede configurarse además para realizar cualquiera de las etapas del método 900.

El sistema 1100 también incluye un decodificador de vídeo 1110. El decodificador de vídeo 1110 comprende un módulo de recepción 1111 para recibir un flujo de bits que comprende un SPS y una imagen codificada, en donde el SPS incluye un sps_video_parameter_set_id, y en donde la imagen codificada está en un conjunto de unidades de NAL de VCL asociadas con una capa que tiene un nuh_layer_id. El decodificador de vídeo 1110 comprende además un módulo de configuración 1113, para ajustar un GeneralLayerIdx[nuh_layer_id] igual a cero cuando el sps_video_parameter_set_id es igual a cero. El decodificador de vídeo 1110 comprende además un módulo de decodificación 1115 para decodificar la imagen codificada de las unidades de NAL de VCL basándose en el GeneralLayerIdx[nuh_layer_id] para producir una imagen decodificada. El decodificador de vídeo 1110 comprende además un módulo de reenvío 1117 para reenviar la imagen decodificada para su visualización como parte de una secuencia de vídeo decodificada. El decodificador de vídeo 1110 puede estar configurado además para realizar cualquiera de los pasos del método 1000.

Un primer componente está acoplado directamente a un segundo componente cuando no hay componentes intervinientes, excepto por una línea, una traza u otro medio, entre el primer componente y el segundo componente. El primer componente está acoplado indirectamente a un segundo componente cuando hay componentes intervinientes distintos de una línea, una traza u otro medio, entre el primer componente y el segundo componente. El término “acoplado” y sus variantes incluye tanto el acoplamiento directo como el acoplamiento indirecto. El uso del término “aproximadamente” significa un rango que incluye ±10% del número siguiente, a menos que se indique lo contrario.

También debe comprenderse que las etapas de los métodos ejemplares expuestos en la presente memoria no necesariamente deben realizarse en el orden descrito, y el orden de las etapas de tales métodos debe comprenderse como meramente ejemplar. Asimismo, se pueden incluir etapas adicionales en tales métodos, y ciertas etapas se pueden omitir o combinar, en métodos coherentes con diversas realizaciones de la presente divulgación.

Si bien en la presente divulgación se han proporcionado varias realizaciones, se puede comprender que los sistemas y métodos divulgados podrían incorporarse en muchas otras formas específicas. Los presentes ejemplos deben considerarse como ilustrativos y no restrictivos, y la intención no es limitarse a los detalles proporcionados en la presente memoria. Por ejemplo, los diversos elementos o componentes podrían estar combinados o integrados en otro sistema, o ciertas características podrían omitirse o no implementarse.

El alcance de la invención está definido por las reivindicaciones adjuntas.

REIVINDICACIONES

1. Un método implementado por un decodificador, comprendiendo el método:

recibir (1001), por el decodificador, un subflujo de bits que comprende un conjunto de parámetros de secuencia, SPS, y una imagen codificada, en donde el SPS incluye un identificador de conjunto de parámetros de vídeo de SPS, sps_video_parameter_set_id, en donde la imagen codificada está en un conjunto de capas de codificación de vídeo, VCL, unidades de capa de abstracción de red, NAL, asociadas con una capa que tiene un identificador de capa de cabecera de unidad de NAL, nuh_layer_id, y en donde el subflujo de bits se obtiene mediante un proceso de extracción de subflujo de bits realizado en un flujo de bits multicapa, siendo el proceso de extracción de subflujo de bits un proceso mediante el cual las unidades de NAL en el flujo de bits multicapa que no pertenecen a un conjunto de objetivos especificado se eliminan del flujo de bits multicapa de tal manera que el subflujo de bits comprende las unidades de NAL que pertenecen al conjunto objetivo;

en donde el sps_video_parameter_set_id especifica un valor de un identificador de conjunto de parámetros de VPS, vps_video_parameter_set_id, cuando es mayor que cero; el SPS no hace referencia a un VPS y no se hace referencia a ningún VPS cuando se decodifica cada secuencia de vídeo de capa codificada que hace referencia al SPS cuando el sps_video_parameter_set_id es igual a cero;

ajustar (1003), por parte del decodificador, un índice de capa general correspondiente a nuh_layer_id, GeneralLayerIdx[nuh_layer_id], igual a cero cuando el sps_video_parameter_set_id es igual a cero, en donde GeneralLayerIdx[nuh_layer_id] es igual a un índice de capa actual;

inferir (1005), por el decodificador, que un valor de un indicador de capa independiente del conjunto de parámetros de vídeo, VPS, para el GeneralLayerIdx[nuh_layer_id], vps_independent_layer_flag[GeneralLayerIdx[nuh_layer_id]], es igual a uno cuando el sps_video_parameter_set_id es igual a cero; especificando vps_independent_layer_flag[GeneralLayerIdx[nuh_layer_id]] que la capa con índice GeneralLayerIdx[nuh_layer_id] no utiliza predicción entre capas cuando tiene un valor igual a uno; y

decodificar (1007), por parte del decodificador, la imagen codificada de las unidades de NAL de VCL basándose en el GeneralLayerIdx[nuh_layer_id] para producir una imagen decodificada.

2. El método de la reivindicación 1, en donde el conjunto de unidades de NAL de VCL es parte de la capa cuando el conjunto de unidades de NAL de VCL tienen todas un valor particular de nuh_layer_id.

3. Un dispositivo de codificación de vídeo que comprende:

un procesador (830), un receptor (810) acoplado al procesador (830), una memoria (832) acoplada al procesador (830), y un transmisor (810) acoplado al procesador (830), en donde el procesador (830), el receptor (810), la memoria (832) y el transmisor (810) están configurados para realizar el método de cualquiera de las reivindicaciones 1 y 2.

4. Un medio no transitorio legible por ordenador, que comprende un producto de programa informático para uso por un dispositivo de codificación de vídeo, comprendiendo el producto de programa informático instrucciones ejecutables por ordenador almacenadas en el medio no transitorio legible por ordenador, de tal manera que cuando son ejecutadas por un procesador hacen que el dispositivo de codificación de vídeo realice el método de cualquiera de las reivindicaciones 1 y 2.

5. Un decodificador, que comprende:

un medio de recepción (1111) para recibir un subflujo de bits que comprende un conjunto de parámetros de secuencia, SPS, y una imagen codificada, en donde el SPS incluye un identificador de conjunto de parámetros de vídeo de SPS, sps_video_parameter_set_id, y en donde la imagen codificada está en un conjunto de unidades de capa de codificación de vídeo, VCL, capa de abstracción de red, NAL, asociadas con una capa que tiene un identificador de capa de cabecera de unidad NAL, nuh_layer_id, y en donde el subflujo de bits se obtiene mediante un proceso de extracción de subflujo de bits realizado en un flujo de bits multicapa, siendo el proceso de extracción de subflujo de bits un proceso mediante el cual las unidades de NAL en el flujo de bits multicapa que no pertenecen a un conjunto objetivo especificado se eliminan del flujo de bits multicapa de tal manera que el subflujo de bits comprende las unidades de NAL que pertenecen al conjunto objetivo;

en donde el sps_video_parameter_set_id especifica un valor de un identificador de conjunto de parámetros de VPS, vps_video_parameter_set_id, cuando es mayor que cero; el SPS no hace referencia a un VPS y no se hace referencia a ningún VPS cuando se decodifica cada secuencia de vídeo de capa codificada que hace referencia al SPS cuando el sps_video_parameter_set_id es igual a cero;

un medio de ajuste (1113) para ajustar un índice de capa general correspondiente al nuh_layer_id, GeneralLayerIdx[nuh_layer_id], a cero cuando el sps_video_parameter_set_id es igual a cero, donde GeneralLayerIdx[nuh_layer_id] es igual a un índice de capa actual; y

un medio de decodificación (1115), para decodificar la imagen codificada de las unidades de NAL de VCL basándose en el GeneralLayerIdx[nuh_layer_id] para producir una imagen decodificada,

y estando configurado el decodificador para inferir que un valor de un indicador de capa independiente del conjunto de parámetros de video, VPS, para GeneralLayerIdx[nuh_layer_id], vps_independent_layer_flag[GeneralLayerIdx[nuh_layer_id]], es igual a uno cuando el sps_video_parameter_set_id es igual a cero; especificando el vps_independent_layer_flag[GeneralLayerIdx[nuh_layer_id]] que la capa con índice GeneralLayerIdx[nuh_layer_id] no usa predicción entre capas cuando tiene un valor igual a uno.

6. El decodificador de la reivindicación 5, en donde el decodificador está configurado además para realizar el método de cualquiera de las reivindicaciones 1 y 2.

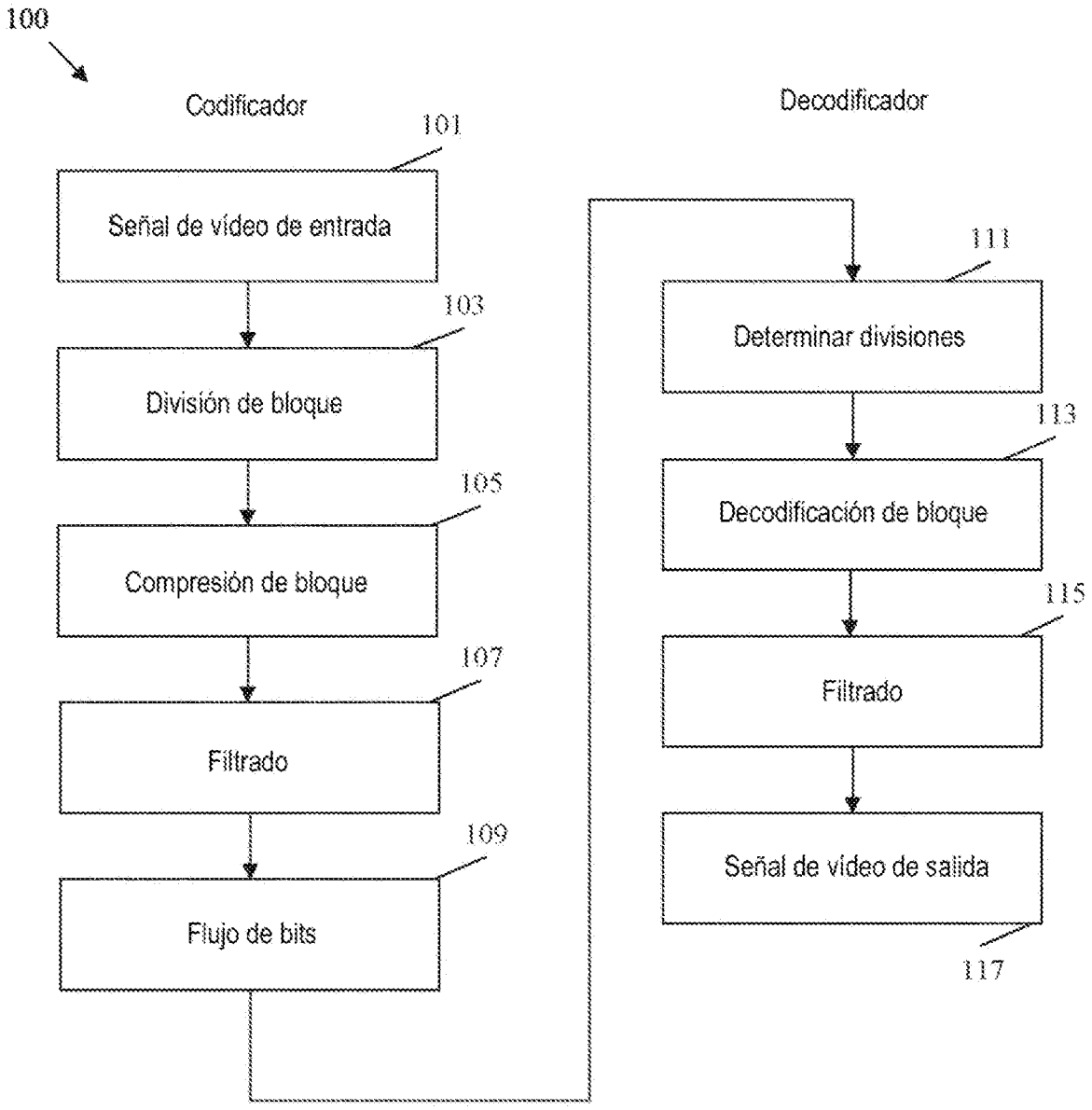
100
Codificador
101
Señal de vídeo de entrada
103
División de bloque
105
Compresión de bloque
107
Filtrado
109
Flujo de bits
Decodificador
111
Determinar divisiones
113
Decodificación de bloque
115
Filtrado
Señal de vídeo de salida
117

FIG. 1

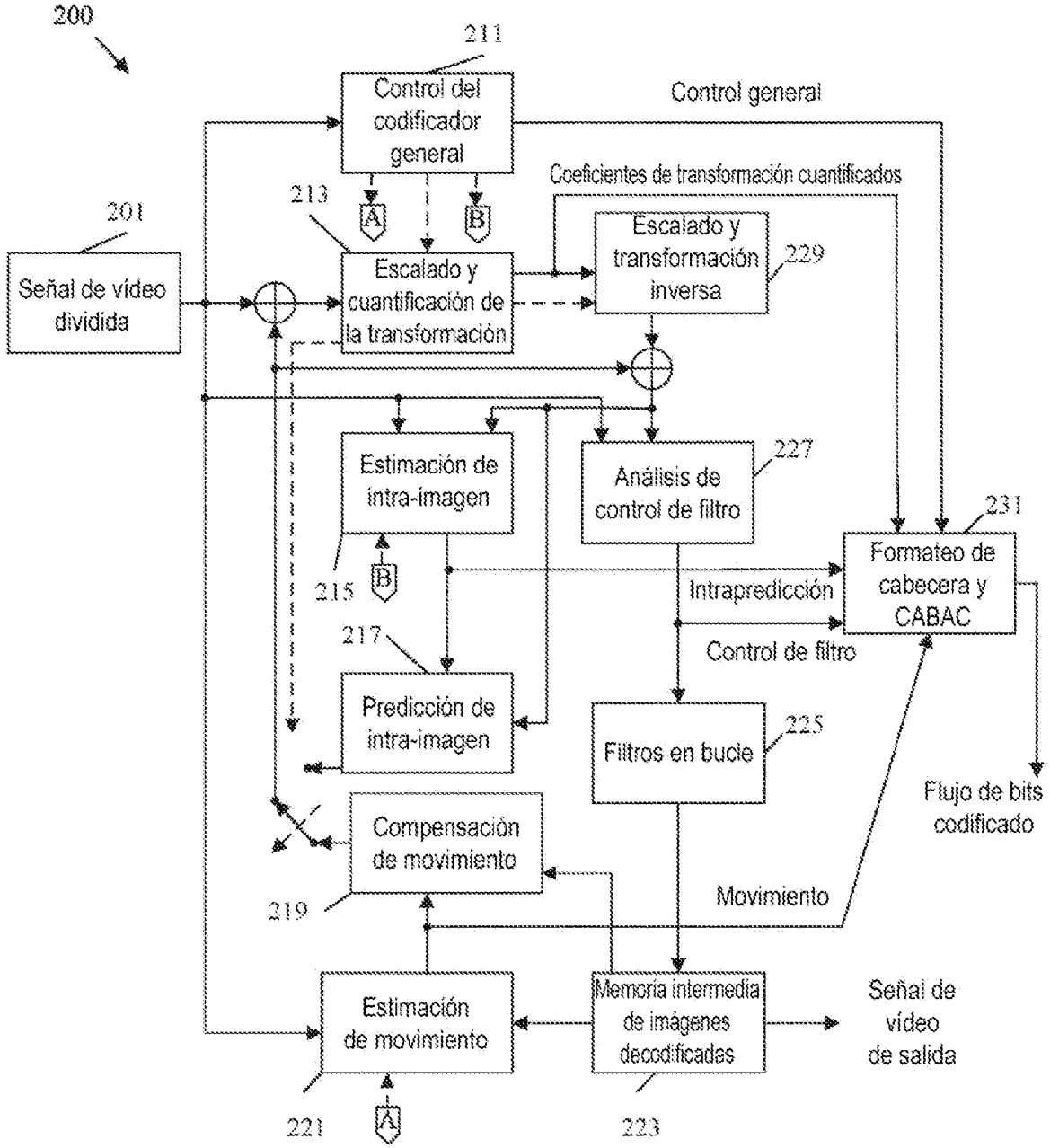
200
211
Control del codificador general
Control general
Coeficientes de transformación cuantificados
213
201
Señal de vídeo dividida
Escalado y cuantificación de la transformación
Escalado y transformación inversa
229
Estimación de intra-imagen
Análisis de control de filtro
227
231
Formateo de cabecera y CABAC
215
Intrapredicción
Control de filtro
217
Predicción de intra-imagen
Filtros en bucle
225
Flujo de bits codificado
Compensación de movimiento
219
Movimiento
Estimación de movimiento
Memoria intermedia de imágenes decodificadas
Señal de vídeo de salida
221
223

FIG. 2

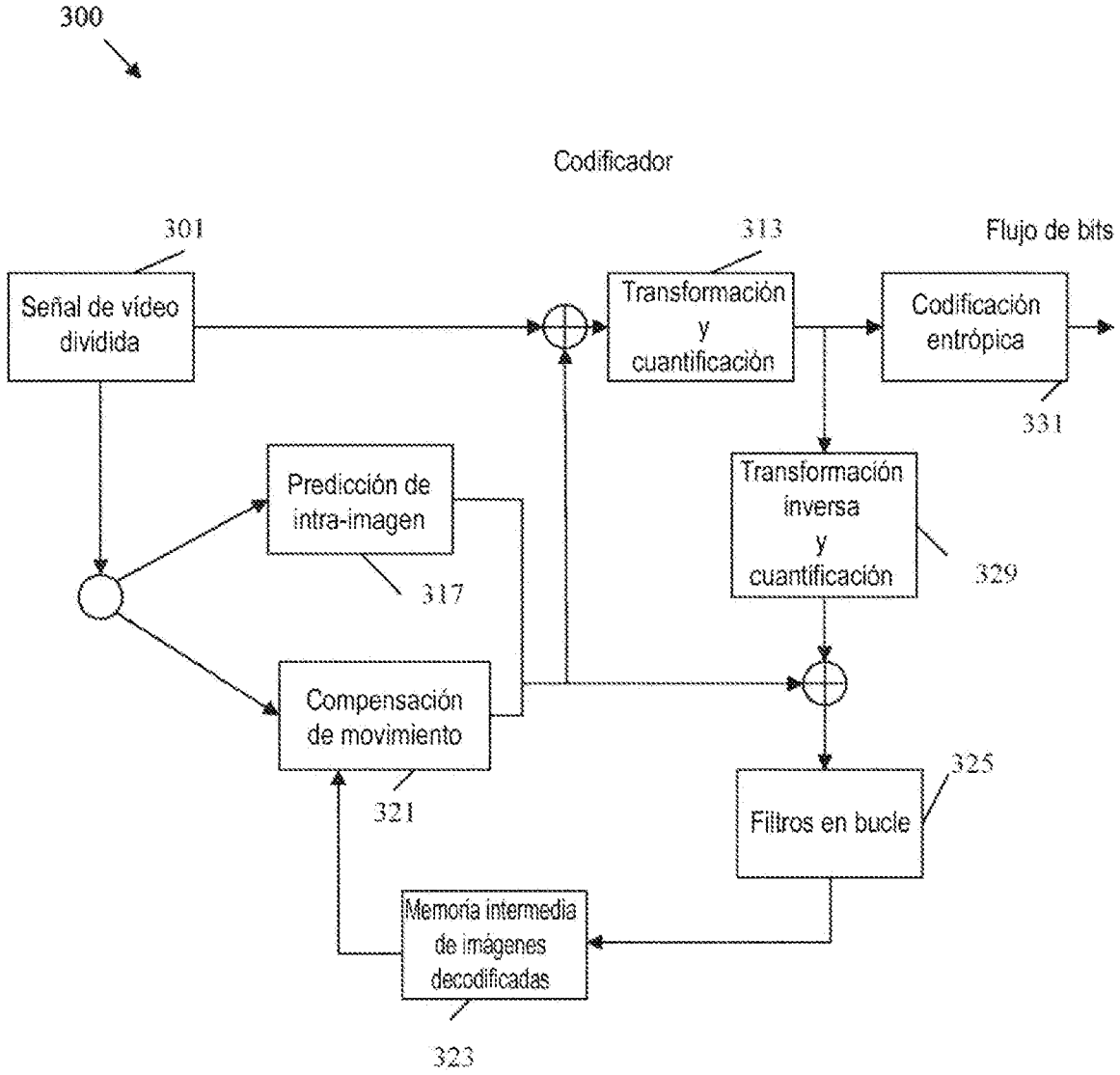
300
Codificador
301
Señal de vídeo dividida
313
Transformación y cuantificación
Flujo de bits
Codificación entrópica
331
Predicción de intra-imagen
317
Transformación inversa y cuantificación
329
Compensación de movimiento
321
325
Filtros en bucle
Memoria intermedia de imágenes decodificadas
323

FIG. 3

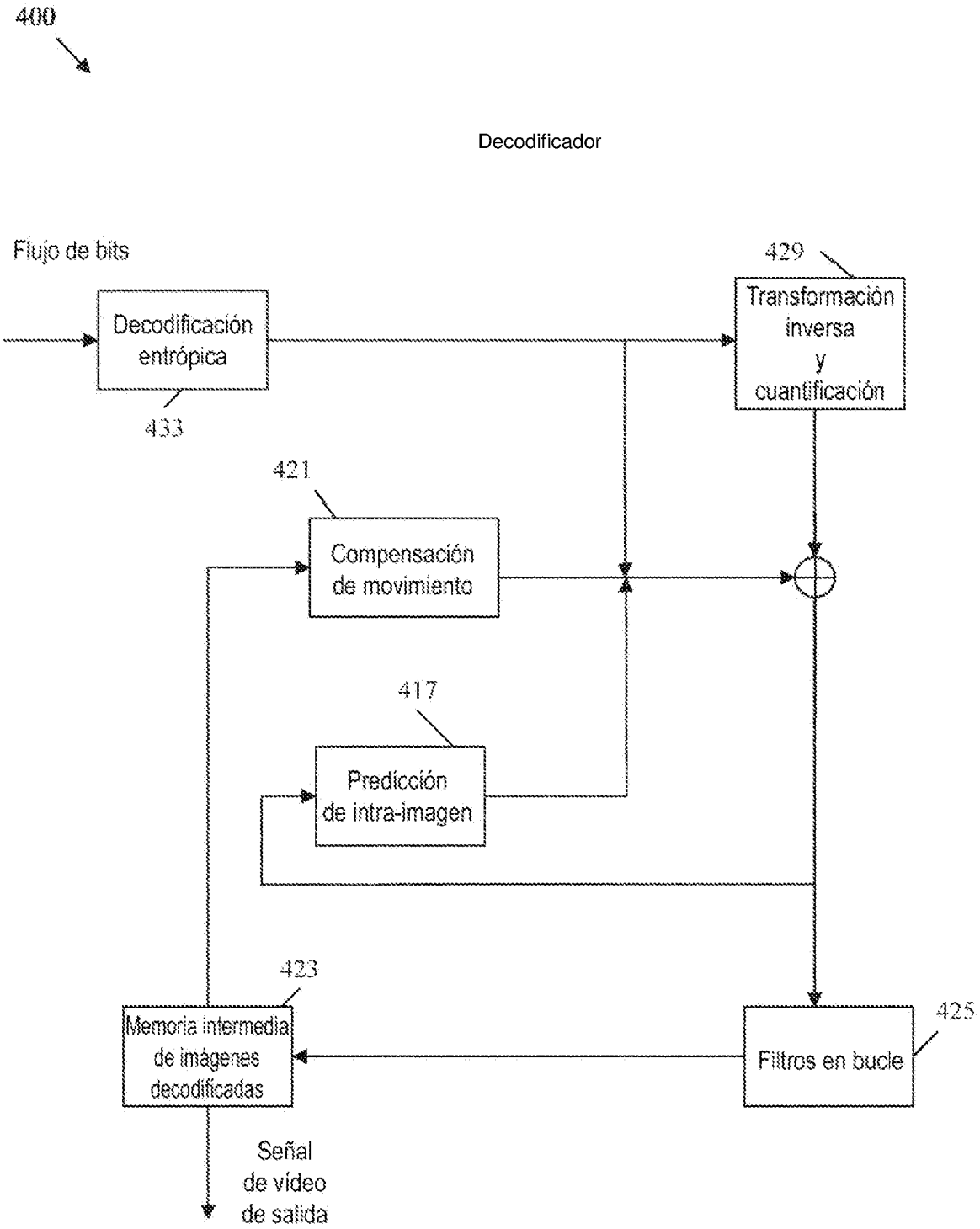
400
Decodificador
Flujo de bits
Decodificación entrópica
433
429
Transformación inversa y cuantificación
421
Compensación de movimiento
417
Predicción de intra-imagen
423
Memoria intermedia de imágenes decodificadas
425
Filtros en bucle
Señal de vídeo de salida

FIG. 4

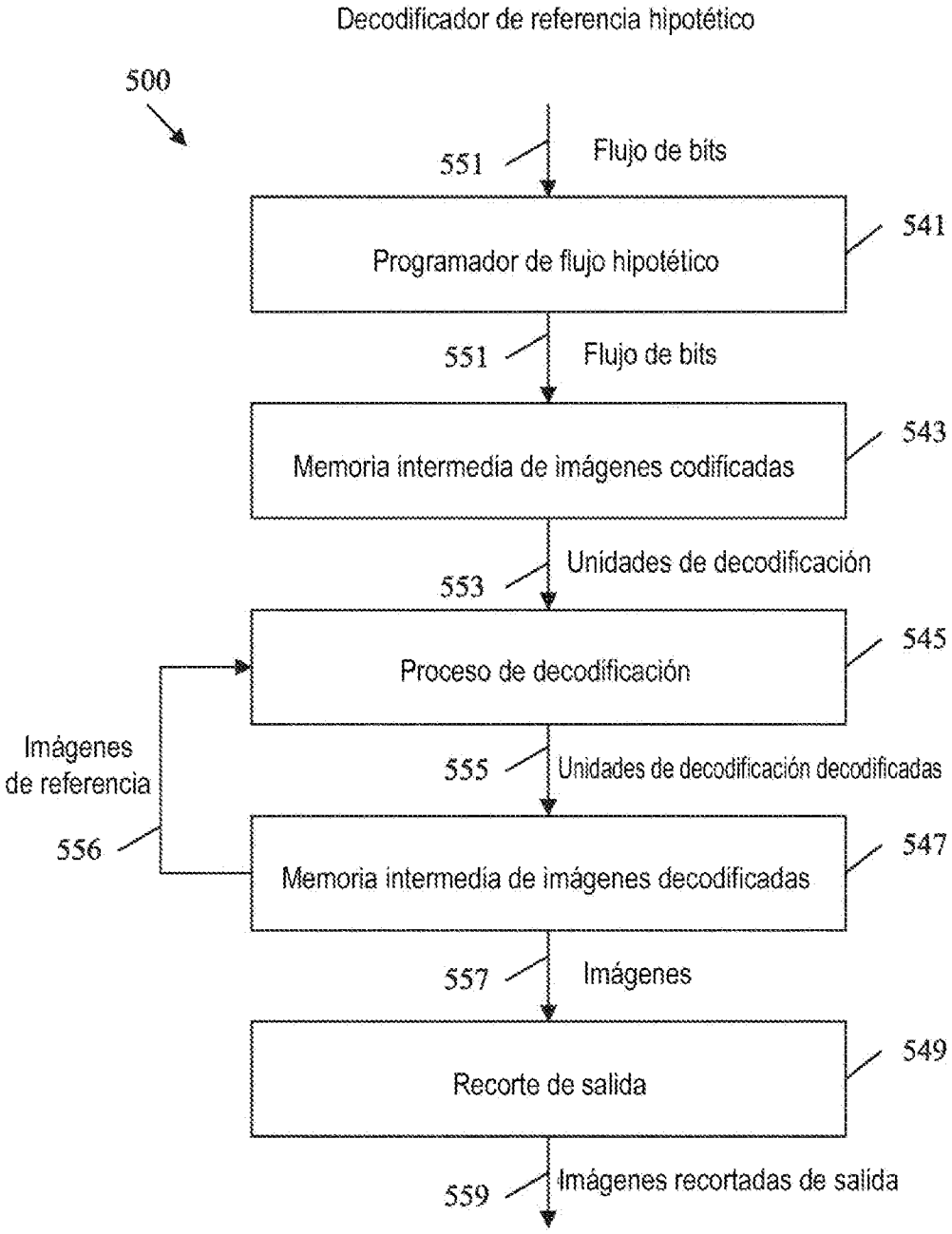
Decodificador de referencia hipotético
500
551
Flujo de bits
Programador de flujo hipotético
541
551
Flujo de bits
Memoria intermedia de imágenes codificadas
543
553
Unidades de decodificación
Proceso de decodificación
545
Imágenes de referencia
555
Unidades de decodificación decodificadas
556
Memoria intermedia de imágenes decodificadas
547
557
Imágenes
Recorte de salida
549
559
Imágenes recortadas de salida

FIG. 5

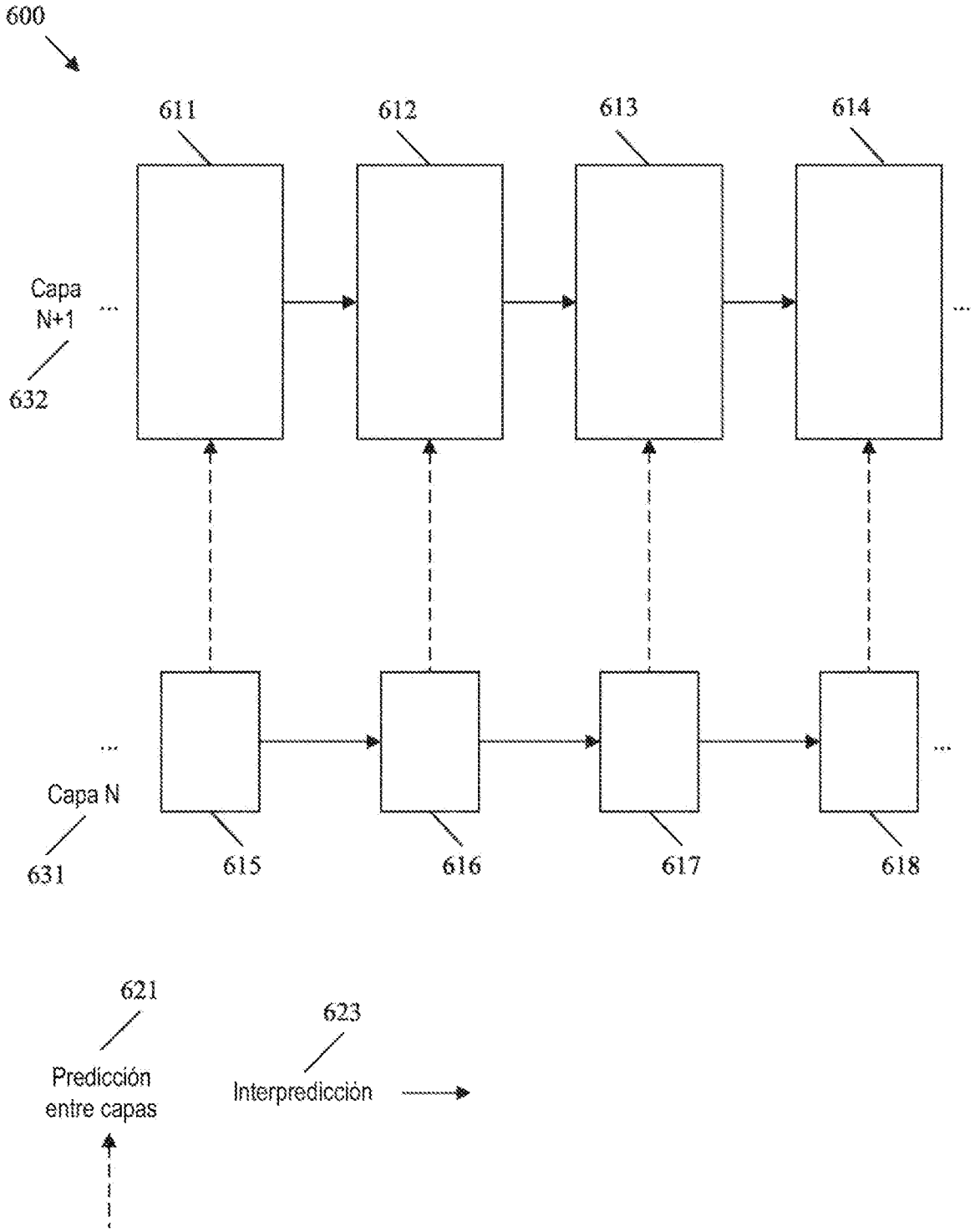
600
611
612
613
614
Capa N+1
632
631
Capa N
615
616
617
618
621
Predicción entre capas
623
Interpredicción

FIG. 6

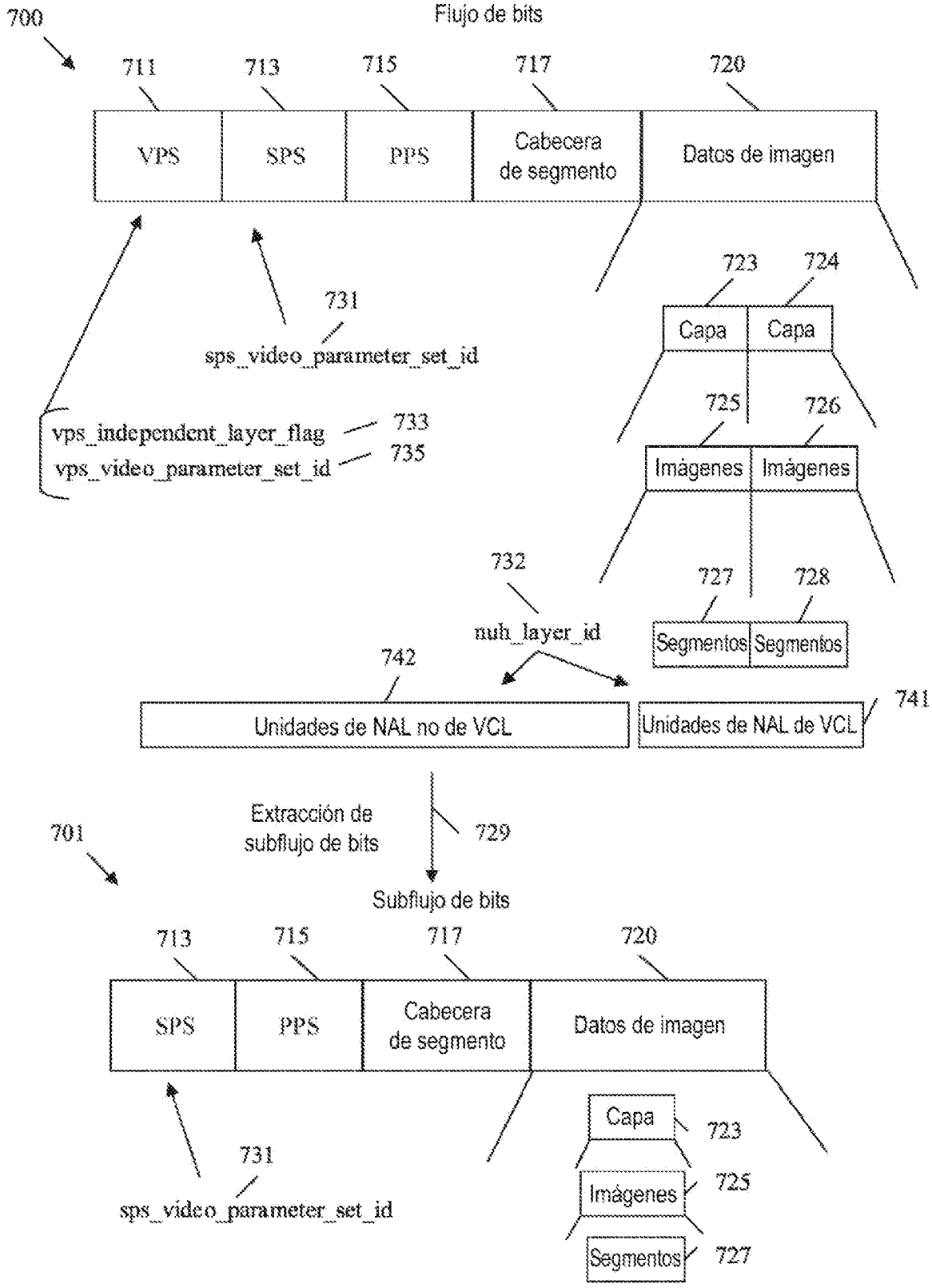
700
Flujo de bits
711
713
715
717
720
VPS
SPS
PPS
Cabecera de segmento
Datos de imagen
731
sps_video_parameter_set_id
vps_independent_layer_flag
733
vps_video_parameter_set_id
735
723
724
Capa
Capa
725
726
Imágenes
Imágenes
727
728
Segmentos
Segmentos
732
nuh_layer_id
742
741
Unidades de NAL no de VCL
Unidades de NAL de VCL
Extracción de subflujo de bits
729
701
Subflujo de bits
713
715
717
720
SPS
PPS
Cabecera de segmento
Datos de imagen
731
sps_video_parameter_set_id
Capa
723
Imágenes
725
Segmentos
727

FIG. 7

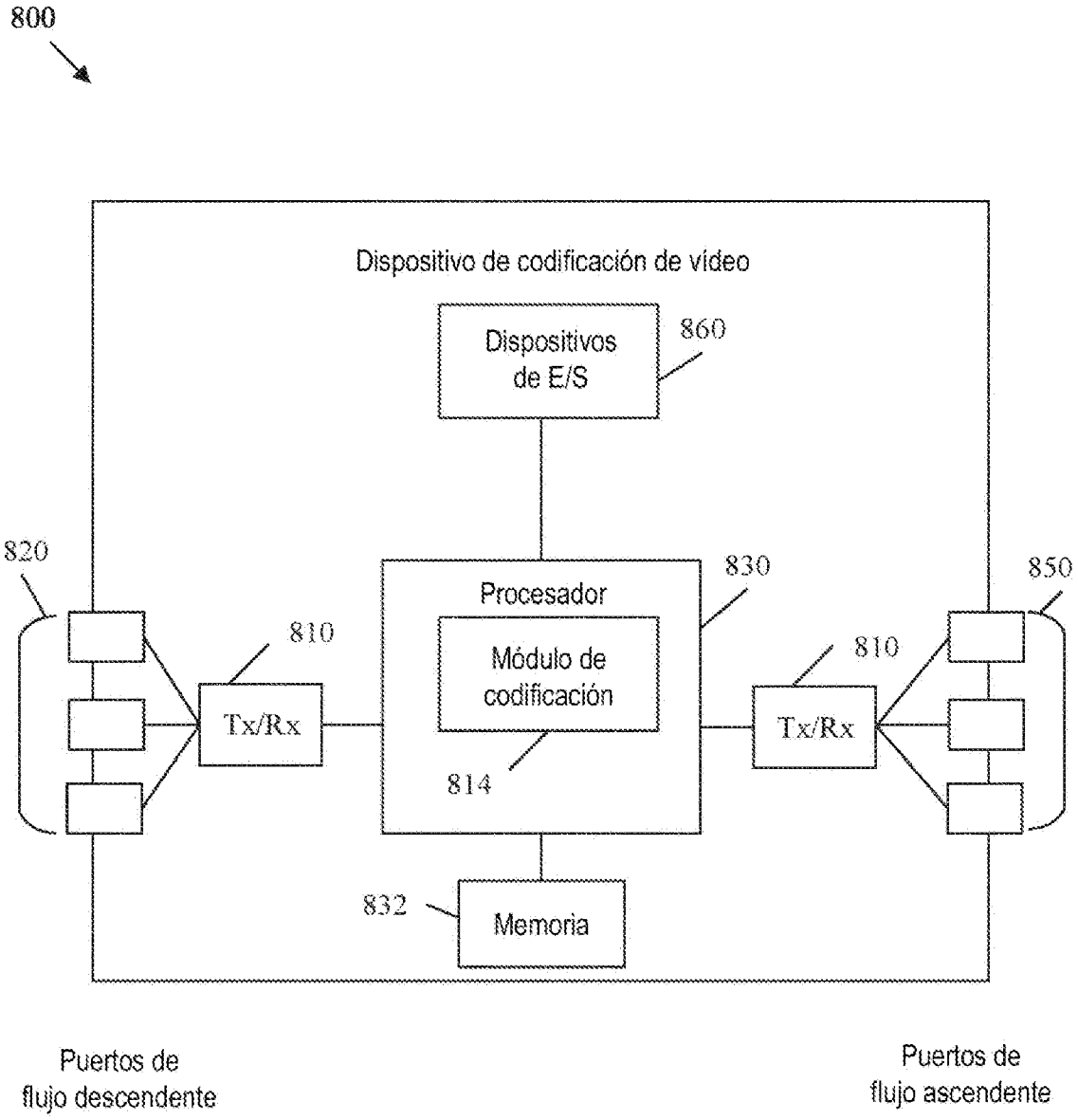
800
Dispositivo de codificación de vídeo
Dispositivos de E/S
860
820
830
850
Procesador
Módulo de codificación
810
Tx/Rx
814
832
Memoria
Puertos de flujo descendente
Puertos de flujo ascendente

FIG. 8

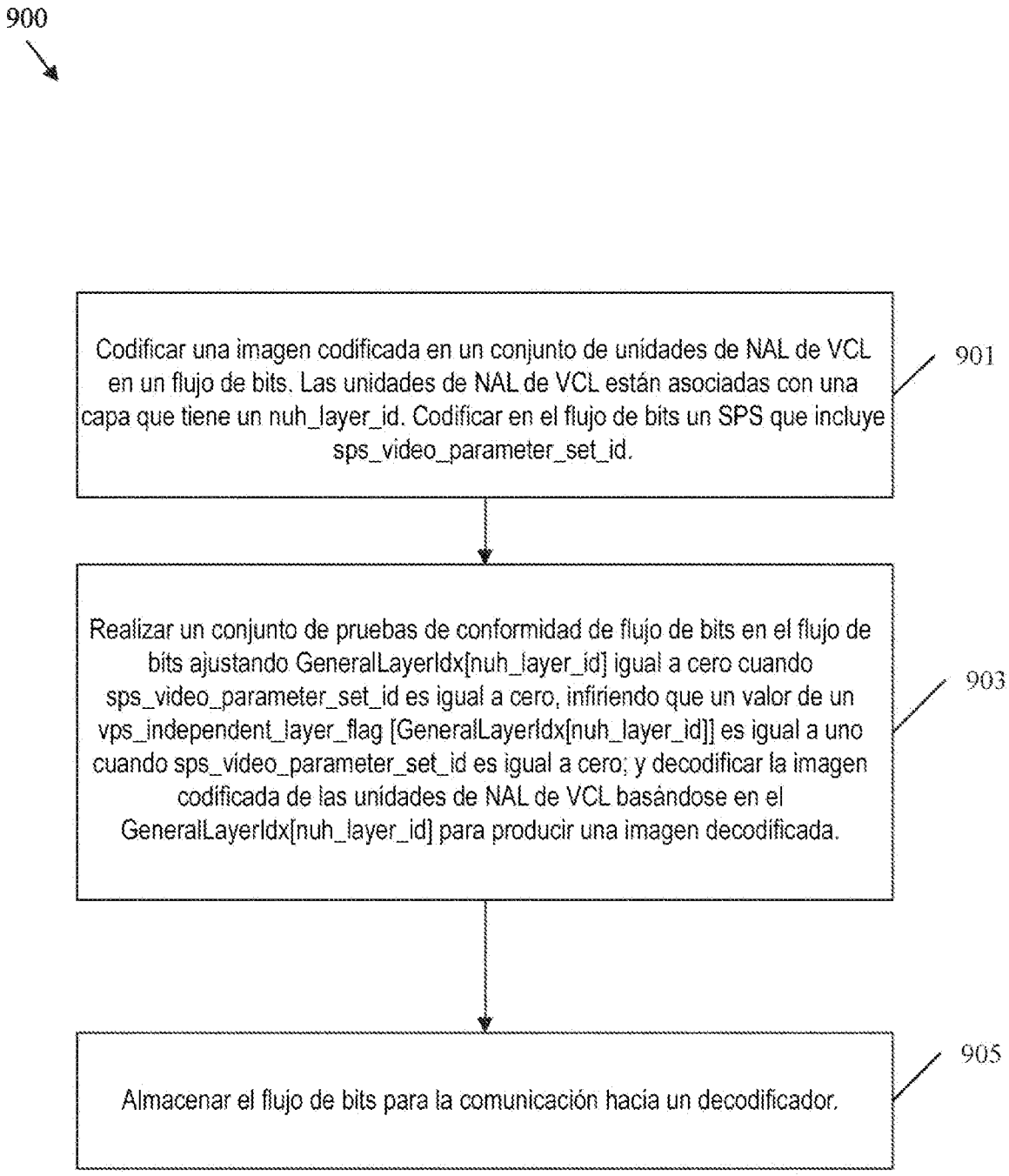
900
Codificar una imagen codificada en un conjunto de unidades de NAL de VCL en un flujo de bits. Las unidades de NAL de VCL están asociadas con una capa que tiene un nuh_layer_id. Codificar en el flujo de bits un SPS que incluye sps_video_parameter_set_id.
901
Realizar un conjunto de pruebas de conformidad de flujo de bits en el flujo de bits ajustando GeneralLayerIdx[nuh_layer_id] igual a cero cuando sps_video_parameter_set_id es igual a cero, infiriendo que un valor de un vps_independent_layer_flag [GeneralLayerIdx[nuh_layer_id]] es igual a uno cuando sps_video_parameter_set_id es igual a cero; y decodificar la imagen codificada de las unidades de NAL de VCL basándose en el GeneralLayerIdx[nuh_layer_id] para producir una imagen decodificada.
903
Almacenar el flujo de bits para la comunicación hacia un decodificador.
905

FIG. 9

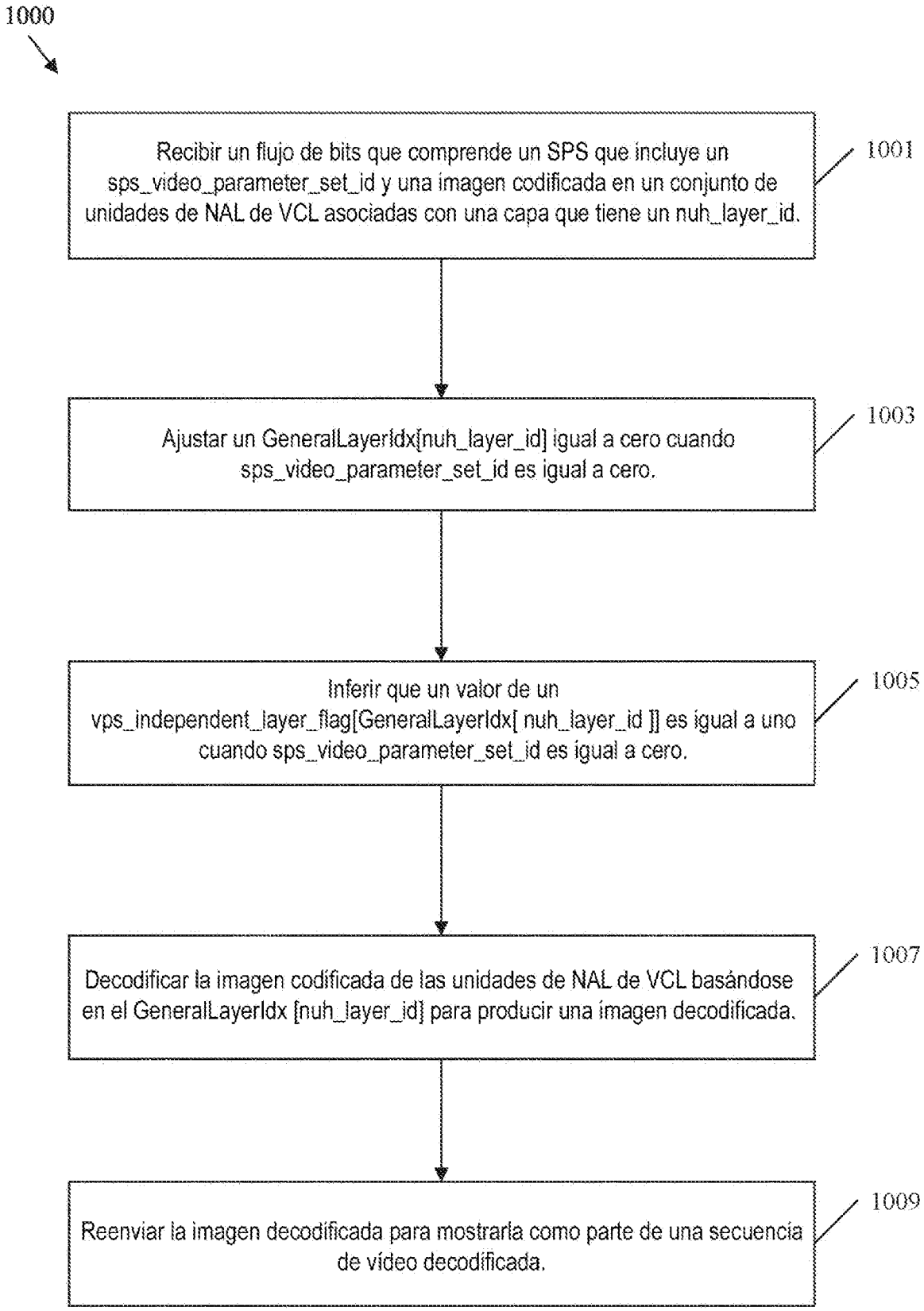
1000
Recibir un flujo de bits que comprende un SPS que incluye un sps_video_parameter_set_id y una imagen codificada en un conjunto de unidades de NAL de VCL asociadas con una capa que tiene un nuh_layer_id.
1001
Ajustar un GeneralLayerIdx[nuh_layer_id] igual a cero cuando sps_video_parameter_set_id es igual a cero.
1003
Inferir que un valor de un vps_independent_layer_flag[GeneralLayerIdx[nuh_layer_id]] es igual a uno cuando sps_video_parameter_set_id es igual a cero.
1005
Decodificar la imagen codificada de las unidades de NAL de VCL basándose en el GeneralLayerIdx [nuh_layer_id] para producir una imagen decodificada.
1007
Reenviar la imagen decodificada para mostrarla como parte de una secuencia de vídeo decodificada.
1009

FIG. 10

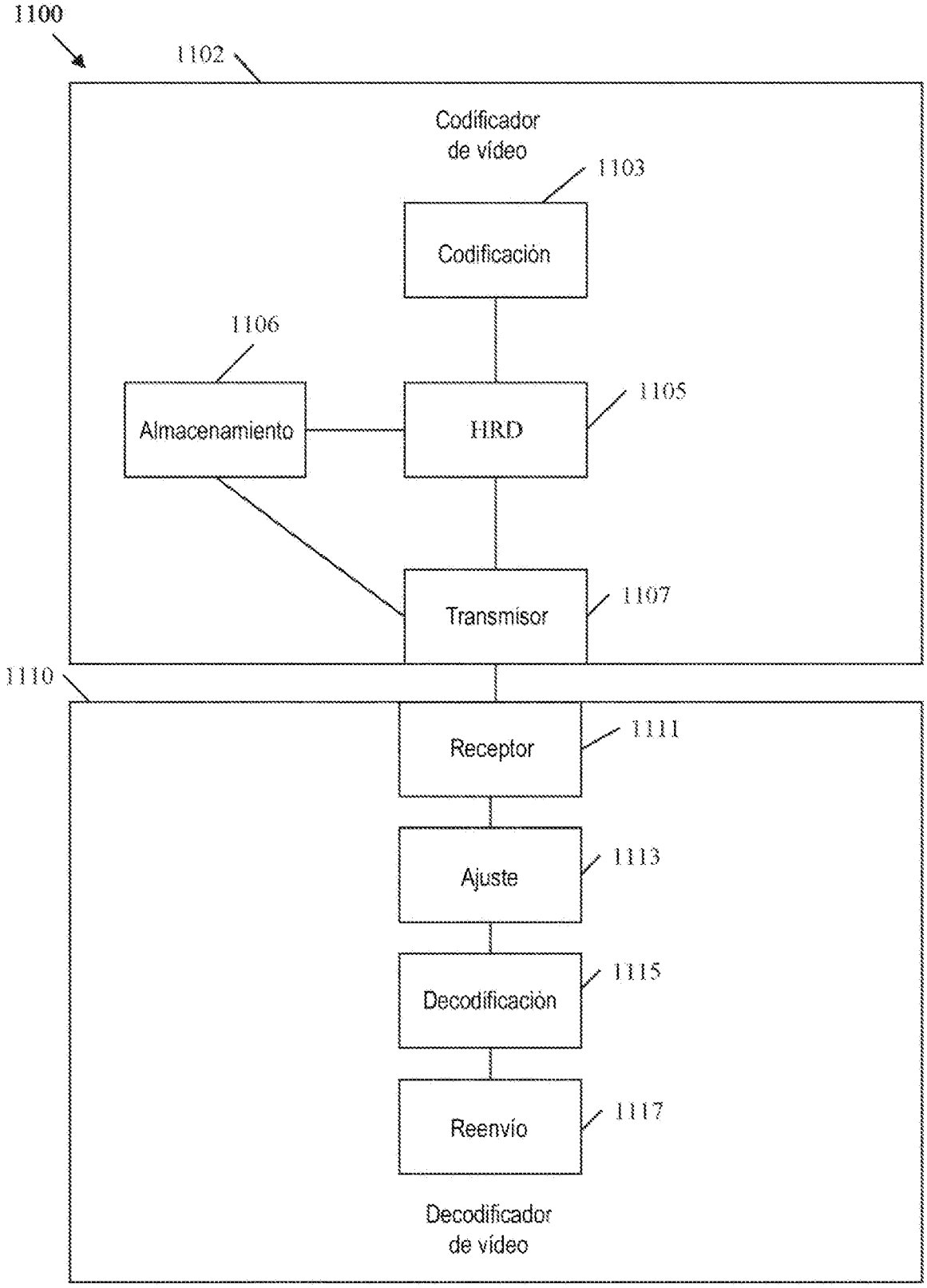
1100
1102
Codificador de vídeo
1103
Codificación
1106
Almacenamiento
HRD
1105
Transmisor
1107
1110
Receptor
1111
Ajuste
1113
Decodificación
1115
Reenvío
1117
Decodificador de vídeo

FIG. 11