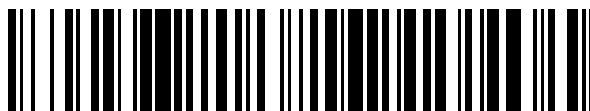


19



OFICINA ESPAÑOLA DE
PATENTES Y MARCAS

ESPAÑA



11 Número de publicación: **2 878 143**

51 Int. Cl.:

H04N 19/70 (2014.01)

H04N 19/46 (2014.01)

H04N 19/30 (2014.01)

12

TRADUCCIÓN DE PATENTE EUROPEA

T3

86 Fecha de presentación y número de la solicitud internacional: **20.11.2013** **PCT/JP2013/006828**

87 Fecha y número de publicación internacional: **30.05.2014** **WO14080630**

96 Fecha de presentación y número de la solicitud europea: **20.11.2013** **E 13856204 (6)**

97 Fecha y número de publicación de la concesión europea: **26.05.2021** **EP 2923488**

54 Título: **Información de escalabilidad de señalización en un conjunto de parámetros**

30 Prioridad:

21.11.2012 US 201261728997 P

04.01.2013 US 201361749173 P

45 Fecha de publicación y mención en BOPI de la
traducción de la patente:
18.11.2021

73 Titular/es:

DOLBY INTERNATIONAL AB (100.0%)
Apollo Building, 3E, Herikerbergweg 1-35
1101 CN Amsterdam Zuidoost, NL

72 Inventor/es:

DESHPANDE, SACHIN G.

74 Agente/Representante:

LINAGE GONZÁLEZ, Rafael

ES 2 878 143 T3

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

DESCRIPCIÓN

Información de escalabilidad de señalización en un conjunto de parámetros

5 Técnica anterior

La presente invención se refiere a la decodificación de video.

10 Los dispositivos electrónicos se han vuelto más pequeños y más potentes para satisfacer las necesidades del consumidor y mejorar la portabilidad y la conveniencia. Los consumidores se han vuelto dependientes de los dispositivos electrónicos y esperan una mayor funcionalidad.

Algunos ejemplos de dispositivos electrónicos incluyen computadoras de escritorio, computadoras portátiles, teléfonos celulares, teléfonos inteligentes, reproductores multimedia, circuitos integrados, etc.

15 Algunos dispositivos electrónicos se usan para procesar y/o mostrar medios digitales. Por ejemplo, los dispositivos electrónicos portátiles ahora permiten producir y/o consumir medios digitales en casi cualquier lugar donde pueda estar un consumidor. Además, algunos dispositivos electrónicos pueden proporcionar descarga o transmisión de contenido de medios digitales para el uso y disfrute de un consumidor.

20 El video digital típicamente se representa como una serie de imágenes o tramas, cada una de las cuales contiene un conjunto de píxeles. Cada píxel incluye información, como información de intensidad y/o color. En muchos casos, cada píxel se representa como un conjunto de tres colores. Algunas técnicas de codificación de video proporcionan una mayor eficiencia de codificación a expensas de una complejidad creciente. El aumento de los requisitos de
25 calidad de imagen y el aumento de los requisitos de resolución de imagen para las técnicas de codificación de video también aumentan la complejidad de la codificación.

La creciente popularidad de los medios digitales ha presentado varios problemas. Por ejemplo, representar de manera eficiente medios digitales de alta calidad para almacenamiento, transmisión y reproducción presenta varios
30 desafíos. Las técnicas que representan los medios digitales de manera más eficiente son beneficiosas.

Los objetivos, características y ventajas anteriores y otros de la invención se entenderán más fácilmente al considerar la siguiente descripción detallada de la invención, tomada junto con los dibujos adjuntos.

35 BOYCE J ET AL: "Diseños de conjuntos de parámetros y encabezado de unidad NAL para extensiones HEVC", 11. REUNIÓN JCT-VC; 102. REUNIÓN MPEG; 10-10-2012-19-10-2012; SHANGHAI; (EQUIPO CONJUNTO COLABORATIVO SOBRE CODIFICACIÓN DE VÍDEO DE ISO/IEC JTC1/SC29/WG11 E ITU-T SG.16); URL: HTTP:// WFTF3. ITU.INT/AV-ARCH/JCTVC-SITE/, no. JCTVC-K1007, 19 de octubre de 2012 (19-10-2012), XP030113273 han divulgado un método para mapear la información proporcionada por un elemento de sintaxis incluido en el
40 encabezado de la unidad NAL a identificadores de dimensión de escalabilidad específicos.

Sumario de la invención

45 La presente invención se refiere a un método para decodificar un flujo de bits de vídeo escalable codificado como se establece en la reivindicación 1. La divulgación que permite la invención se encuentra en relación con las realizaciones de las figuras 32 a 39.

Las realizaciones/ejemplos restantes deben entenderse como ejemplos que son útiles para la comprensión y se proporcionan solo con fines ilustrativos.

50 Breve descripción de los dibujos

[fig.1] La figura 1 es un diagrama de bloques que ilustra una configuración de un dispositivo electrónico que incluye un codificador HEVC.

55 [fig.2] La figura 2 es un diagrama de bloques que ilustra una configuración de un dispositivo electrónico que incluye un decodificador HEVC.

[fig.3] La figura 3 es un diagrama de bloques que ilustra un ejemplo de un codificador y un decodificador.

60 [fig.4] La figura 4 ilustra varios componentes que pueden utilizarse en un dispositivo electrónico.

[fig.5] La figura 5 ilustra una estructura de segmentos de ejemplo.

65 [fig. 6] La figura 6 ilustra otra estructura de segmentos de ejemplo.

[fig. 7] La figura 7 ilustra una trama con un segmento y 9 mosaicos.

[fig.8] La figura 8 ilustra una trama con tres segmentos y 3 mosaicos.

5 [fig. 9A] La figura 9A ilustra una sintaxis de encabezado de unidad NAF diferente.

[fig. 9B] La figura 9B ilustra una sintaxis de encabezado de unidad NAF diferente.

[fig. 9C] La figura 9C ilustra una sintaxis de encabezado de unidad NAF diferente.

10 [fig. 10] La figura 10 ilustra una sintaxis de unidad NAF general.

[fig.11] La figura 11 ilustra un conjunto de parámetros de vídeo existente.

15 [fig. 12] La figura 12 ilustra los tipos de escalabilidad existentes.

[fig. 13] La figura 13 ilustra un conjunto de parámetros de vídeo de ejemplo.

[fig. 14] La figura 14 ilustra una sintaxis de mapa de escalabilidad de ejemplo.

20 [fig.15] La figura 15 ilustra un conjunto de parámetros de vídeo de ejemplo.

[fig. 16] La figura 16 ilustra un conjunto de parámetros de vídeo existente.

25 [fig.17] La figura 17 ilustra un tipo de dimensión existente, sintaxis de ID de dimensión.

[fig.18] La figura 18 ilustra un conjunto de parámetros de vídeo de ejemplo.

[fig.19] La figura 19 ilustra una sintaxis de mapa de escalabilidad de ejemplo.

30 [fig.20] La figura 20 ilustra un conjunto de parámetros de vídeo de ejemplo.

[fig.21] La figura 21 ilustra un conjunto de parámetros de vídeo de ejemplo.

35 [fig.22] La figura 22 ilustra un conjunto de parámetros de vídeo de ejemplo.

[fig.23] La figura 23 ilustra una sintaxis de indicador de información de tipo de escaneo de fuente de ejemplo.

[fig.24] La figura 24 ilustra una sintaxis de indicador de información de fuente de ejemplo.

40 [fig.25] La figura 25 ilustra un conjunto de parámetros de vídeo de ejemplo.

[fig. 26] La figura 26 ilustra un conjunto de parámetros de vídeo de ejemplo.

45 [fig. 27] La figura 27 ilustra una sintaxis de indicador de información de tipo de escaneo de fuente de ejemplo.

[fig.28] La figura 28 ilustra una sintaxis de indicador de información de fuente de ejemplo.

[fig.29] La figura 29 ilustra una sintaxis de mapa de escalabilidad de ejemplo.

50 [fig.30] La figura 30 ilustra un conjunto de parámetros de vídeo de ejemplo.

[fig.31] La figura 31 ilustra una sintaxis de máscara de escalabilidad de ejemplo.

55 [fig. 32] La figura 32 ilustra una sintaxis de extensión de conjunto de parámetros de vídeo de ejemplo.

[fig. 33] La figura 33 ilustra una sintaxis de extensión de conjunto de parámetros de vídeo de ejemplo.

[fig.34] La figura 34 ilustra una sintaxis de extensión de conjunto de parámetros de vídeo de ejemplo.

60 [fig. 35] La figura 35 ilustra una sintaxis de extensión de conjunto de parámetros de vídeo de ejemplo.

[fig. 36] La figura 36 ilustra una sintaxis de extensión de conjunto de parámetros de vídeo de ejemplo.

65 [fig. 37] La figura 37 ilustra una sintaxis de extensión de conjunto de parámetros de vídeo de ejemplo.

[fig. 38] La figura 38 ilustra una sintaxis de extensión de conjunto de parámetros de vídeo de ejemplo.

[fig. 39] La figura 39 ilustra una sintaxis de extensión de conjunto de parámetros de vídeo de ejemplo.

5 Descripción de realizaciones

El equipo conjunto colaborativo sobre codificación de vídeo (JCT-VC) de la comisión de estudio 16 (SG16) del sector de normalización de telecomunicaciones de la Unión Internacional de Telecomunicaciones (ITU-T) grupo de trabajo 3 (WP3) y la Organización Internacional de Normalización/Comisión Electrotécnica Internacional (ISO/IEC) El comité técnico conjunto 1/subcomité 29/grupo de trabajo 11 (JTC1/SC29/WG11) ha lanzado un esfuerzo de estandarización para un estándar de codificación de vídeo llamado estándar de codificación de vídeo de alta eficiencia (HEVC). HEVC usa codificación basada en bloques.

En HEVC, se usa una técnica de codificación de entropía (CABAC) de codificación aritmética binaria adaptativa al contexto) para comprimir los coeficientes transformados y cuantificados (TQC) sin pérdidas. Los TQC pueden ser de diferentes tamaños de bloque de acuerdo con los tamaños de transformación (por ejemplo, 4x4, 8x8, 16x16, 32x32).

Los TQC bidimensionales (2D) se pueden convertir en una matriz unidimensional (1D) antes de la codificación de entropía. En un ejemplo, los TQC dispuestos en 2D en un bloque 4x4 pueden disponerse como se ilustra en la Tabla (1).

Tabla (1)

4	0		1	0
3	2		-1	...
-3	0		...	...
0	...		...	...

Al convertir los TQC 2D en una matriz de ID, el bloque se puede escanear en forma de zig-zag diagonal. Continuando con el ejemplo, los TQC en matriz 2D ilustrados en la Tabla (1) se pueden convertir en TQC en matriz de ID [4, 0, 3, -3, 2, 1, 0, -1, 0, ...] escaneando la primera fila y la primera columna, la primera fila y la segunda columna, la segunda fila y la primera columna, la tercera fila y la primera columna, la segunda fila y la segunda columna, la primera fila y la tercera columna, la primera fila y la cuarta columna, la segunda fila y la tercera columna, la tercera fila y la segunda columna, la cuarta fila y la primera columna y así sucesivamente.

El procedimiento de codificación en HEVC puede proceder, por ejemplo, de la siguiente manera. Los TQC en la matriz de ID se pueden ordenar de acuerdo con la posición de escaneo. Puede determinarse la posición de escaneo del último coeficiente significativo y el último nivel de coeficiente. Se puede codificar el último coeficiente significativo. Cabe señalar que los coeficientes se codifican típicamente en orden de escaneo inverso. Se puede realizar la codificación a nivel de ejecución, que codifica información sobre las ejecuciones de números y/o bits idénticos en lugar de codificar los números en sí, que se activa directamente después de la última codificación de coeficientes. Entonces, se puede realizar la codificación de nivel. El término coeficiente significativo se refiere a un coeficiente que tiene un valor de nivel de coeficiente que es mayor que cero. Un valor de nivel de coeficiente se refiere a un indicador único de la magnitud (o valor absoluto) de un valor de coeficiente transformado y cuantificado (TQC).

Este procedimiento puede ilustrarse en la Tabla (2) como una continuación del ejemplo anterior (con los TQC en matriz de ID [4, 0, 3, -3, 2, 1, 0, -1, 0, ...]).

Tabla (2)

Posición de escaneo	0	1	2	3	4	5	6	7	...
Nivel de coeficiente	4	0	3	-3	2	1	0	-1	...
Última posición								7	
Último nivel de coeficiente								-1	
Codificación de nivel de ejecución					2	1	0		
Codificación de nivel	4	0	3	-3					

En la tabla (2), por ejemplo, el nivel de coeficiente -1 en la posición de escaneo 7 puede ser el último coeficiente distinto de cero. Por tanto, la última posición es la posición de escaneo 7 y el último nivel de coeficiente es -1. La codificación a nivel de ejecución se puede realizar para los coeficientes 0, 1 y 2 en las posiciones de escaneo 6, 5 y 4 (donde los coeficientes se codifican en orden de escaneo inverso). Luego, se puede realizar la codificación de nivel para los niveles de coeficiente -3, 3, 0 y 4.

La figura 1 es un diagrama de bloques que ilustra una configuración de un dispositivo electrónico 102 en el que se puede codificar vídeo. Cabe señalar que uno o más de los elementos ilustrados como incluidos dentro del dispositivo electrónico 102 pueden implementarse en hardware, software o una combinación de ambos. Por ejemplo, el

dispositivo electrónico 102 incluye un codificador 108, que puede implementarse en hardware, software o una combinación de ambos. Por ejemplo, el codificador 108 puede implementarse como un circuito, circuito integrado, circuito integrado de aplicación específica (ASIC), procesador en comunicación electrónica con memoria con instrucciones ejecutables, firmware, matriz de puerta programable en campo (FPGA), etc., o una combinación de los mismos. En algunas configuraciones, el codificador 108 puede ser un codificador de codificación de video de alta eficiencia (HEVC).

El dispositivo electrónico 102 puede incluir un proveedor 104. El proveedor 104 puede proporcionar imágenes o datos de imágenes (por ejemplo, video) como una fuente 106 al codificador 108. Los ejemplos del proveedor 104 incluyen sensores de imagen, memoria, interfaces de comunicación, interfaces de red, receptores inalámbricos, puertos, etc.

La fuente 106 puede proporcionarse a un módulo de predicción intratrama y una memoria intermedia 110 de reconstrucción. La fuente 106 también se puede proporcionar a un módulo 136 de estimación de movimiento y compensación de movimiento y a un módulo 116 de sustracción.

El módulo de predicción intratrama y la memoria intermedia 110 de reconstrucción pueden generar información intramodo 128 y una intraseñal 112 basándose en la fuente 106 y los datos reconstruidos 150. El módulo 136 de estimación de movimiento y compensación de movimiento puede generar información intermodo 138 y una interseñal 114 basándose en la fuente 106 y una señal 168 de memoria intermedia 166 de imagen de referencia. La señal 168 de la memoria intermedia 166 de imagen de referencia puede incluir datos de una o más imágenes de referencia almacenadas en la memoria intermedia 166 de imagen de referencia.

El codificador 108 puede seleccionar entre la intraseñal 112 y la interseñal 114 de acuerdo con un modo. La intraseñal 112 puede usarse para explotar características espaciales dentro de una imagen en un intramodo de codificación. La interseñal 114 se puede usar para aprovechar las características temporales entre imágenes en un modo de intercodificación. Mientras está en el modo de intracodificación, la intraseñal 112 puede proporcionarse al módulo 116 de sustracción y la información intramodo 128 puede proporcionarse a un módulo 130 de codificación de entropía. Mientras está en el modo de intercodificación, la señal intermodo 114 puede proporcionarse al módulo 116 de sustracción y la información intermodo 138 puede proporcionarse al módulo 130 de codificación de entropía.

O bien la intraseñal 112 o la interseñal 114 (dependiendo del modo) se sustrae de la fuente 106 en el módulo 116 de sustracción con el fin de producir una predicción residual 118. La predicción residual 118 se proporciona a un módulo 120 de transformación. El módulo 120 de transformación puede comprimir el residuo 118 de predicción para producir una señal 122 transformada que se proporciona a un módulo 124 de cuantificación. El módulo de cuantificación 124 cuantifica la señal transformada 122 para producir coeficientes transformados y cuantificados (TQC) 126.

Los TQC 126 se proporcionan a un módulo 130 de codificación de entropía y un módulo 140 de cuantificación inversa. El módulo 140 de cuantificación inversa realiza una cuantificación inversa en los TQC 126 para producir una señal 142 cuantificada inversa que se proporciona a un módulo 144 de transformación inversa. El módulo 144 de transformación inversa descomprime la señal cuantificada inversa 142 para producir una señal descomprimida 146 que se proporciona a un módulo 148 de reconstrucción.

El módulo 148 de reconstrucción puede producir datos reconstruidos 150 basándose en la señal descomprimida 146. Por ejemplo, el módulo 148 de reconstrucción puede reconstruir (modificar) imágenes. Los datos reconstruidos 150 se pueden proporcionar a un filtro 152 de desbloqueo y al módulo de intrapredicción y la memoria intermedia 110 de reconstrucción. El filtro 152 de desbloqueo puede producir una señal 154 filtrada basándose en los datos 150 reconstruidos.

La señal filtrada 154 puede proporcionarse a un módulo 156 de desplazamiento adaptativo de muestra (SAO). El módulo SAO 156 puede producir información SAO 158 que se proporciona al módulo 130 de codificación de entropía y una señal SAO 160 que se proporciona a un filtro 162 de bucle adaptativo (ALF). El ALF 162 produce una señal ALF 164 que se proporciona a la memoria intermedia 166 de imagen de referencia. La señal ALF 164 puede incluir datos de una o más imágenes que pueden usarse como imágenes de referencia. En algunos casos, el ALF 162 puede omitirse.

El módulo 130 de codificación de entropía puede codificar los TQC 126 para producir un flujo 134 de bits. Como se describió anteriormente, los TQC 126 se pueden convertir en una matriz de ID antes de la codificación de entropía. Además, el módulo 130 de codificación de entropía puede codificar los TQC 126 usando CAVLC o CABAC. En particular, el módulo 130 de codificación de entropía puede codificar los TQC 126 basándose en una o más de la información intermodo 128, la información intermodo 138 y la información SAO 158. El flujo 134 de bits puede incluir datos de imagen codificados.

La cuantificación, involucrada en la compresión de video como HEVC, es una técnica de compresión con pérdida que se logra al comprimir un rango de valores a un solo valor cuántico. El parámetro de cuantificación (QP) es un parámetro de escala predefinido que se usa para realizar la cuantificación basándose tanto en la calidad del video

reconstruido como en la relación de compresión. El tipo de bloque se define en HEVC para representar las características de un bloque dado basándose en el tamaño del bloque y su información de color. QP, la información de resolución y el tipo de bloque se pueden determinar antes de la codificación de entropía. Por ejemplo, el dispositivo electrónico 102 (por ejemplo, el codificador 108) puede determinar el QP, la información de resolución y el tipo de bloque, que puede proporcionarse al módulo 130 de codificación de entropía.

El módulo 130 de codificación de entropía puede determinar el tamaño del bloque basándose en un bloque de TQC 126. Por ejemplo, el tamaño del bloque puede ser el número de TQC 126 a lo largo de una dimensión del bloque de TQC. En otras palabras, el número de TQC 126 en el bloque de TQC puede ser igual al cuadrado del tamaño del bloque. Además, el bloque puede no ser cuadrado donde el número de TQC 126 es la altura multiplicada por el ancho del bloque. Por ejemplo, el tamaño del bloque puede determinarse como la raíz cuadrada del número de TQC 126 en el bloque de TQC. La resolución puede definirse como un ancho de píxel por una altura de píxel. La información de resolución puede incluir varios píxeles para el ancho de una imagen, para la altura de una imagen o ambos. El tamaño del bloque puede definirse como el número de TQC a lo largo de una dimensión de un bloque 2D de TQC.

En algunas configuraciones, el flujo 134 de bits puede transmitirse a otro dispositivo electrónico. Por ejemplo, el flujo 134 de bits se puede proporcionar a una interfaz de comunicación, interfaz de red, transmisor inalámbrico, puerto, etc. Por ejemplo, el flujo 134 de bits puede transmitirse a otro dispositivo electrónico a través de una red de área local (LAN), Internet, una estación base de telefonía celular, etc. El flujo 134 de bits puede almacenarse adicional o alternativamente en la memoria del dispositivo electrónico 102.

La figura 2 es un diagrama de bloques que ilustra una configuración de un dispositivo electrónico 270 que incluye un decodificador 272 que puede ser un decodificador de codificación de vídeo de alta eficiencia (HEVC). El decodificador 272 y uno o más de los elementos ilustrados incluidos en el decodificador 272 pueden implementarse en hardware, software o una combinación de ambos. El decodificador 272 puede recibir un flujo 234 de bits (por ejemplo, una o más imágenes codificadas incluidas en el flujo 234 de bits) para decodificar. En algunas configuraciones, el flujo 234 de bits recibido puede incluir información de sobrecarga recibida, como un encabezado de segmento recibido, conjunto de parámetros de imagen recibido (PPS), información de descripción de la memoria intermedia recibida, indicador de clasificación, etc.

Los símbolos recibidos (por ejemplo, TQC codificados) del flujo 234 de bits pueden decodificarse por entropía mediante un módulo 274 de decodificación de entropía. Esto puede producir una señal 298 de información de movimiento y coeficientes transformados y cuantificados 278 (TQC) decodificados.

La señal 298 de información de movimiento puede combinarse con una porción de una imagen 292 decodificada de una memoria 290 de trama en un módulo 294 de compensación de movimiento, que puede producir una señal 296 de predicción intertrama. Los coeficientes transformados y cuantificados (TQC) 278 decodificados pueden cuantificarse y transformarse inversamente mediante un módulo 280 de cuantificación inversa y transformación inversa, produciendo así una señal residual decodificada 282. La señal residual decodificada 282 puede añadirse a una señal 205 de predicción mediante un módulo 207 de suma para producir una señal combinada 284. La señal 205 de predicción puede ser una señal seleccionada entre la señal 296 de predicción intertrama producida por el módulo 294 de compensación de movimiento o una señal 203 de predicción intratrama producida por un módulo 201 de predicción intratrama. En algunas configuraciones, esta selección de señal puede estar basada en (por ejemplo, controlada por) el flujo 234 de bits.

La señal 203 de predicción intratrama puede predecirse a partir de información previamente decodificada de la señal combinada 284 (en la trama actual, por ejemplo). La señal combinada 284 también puede filtrarse mediante un filtro 286 de desbloqueo. La señal filtrada resultante 288 puede proporcionarse a un módulo 231 de desplazamiento adaptativo de muestra (SAO). Basándose en la señal filtrada 288 y la información 239 del módulo 274 de decodificación de entropía, el módulo SAO 231 puede producir una señal SAO 235 que se proporciona a un filtro 233 de bucle adaptativo (ALF). El ALF 233 produce una señal ALF 237 que se proporciona a la memoria 290 de trama. La señal ALF 237 puede incluir datos de una o más imágenes que pueden usarse como imágenes de referencia. La señal ALF 237 puede escribirse en la memoria 290 de trama. La señal ALF resultante 237 puede incluir una imagen decodificada. En algunos casos, se puede omitir el ALF 233.

La memoria 290 de trama puede incluir una memoria intermedia de imágenes decodificadas (DPB). La memoria 290 de trama también puede incluir información general correspondiente a las imágenes decodificadas. Por ejemplo, la memoria 290 de trama puede incluir encabezados de segmento, información de conjunto de parámetros de imagen (PPS), parámetros de ciclo, información de descripción de la memoria intermedia, etc. Una o más de estas informaciones se pueden señalar desde un codificador (por ejemplo, el codificador 108).

La memoria 290 de trama puede proporcionar una o más imágenes decodificadas 292 al módulo 294 de compensación de movimiento. Además, la memoria 290 de trama puede proporcionar una o más imágenes 292 decodificadas, que pueden salir del decodificador 272. Dicha o más imágenes decodificadas 292 pueden presentarse en una pantalla, almacenarse en la memoria o transmitirse a otro dispositivo, por ejemplo.

La figura 3 es un diagrama de bloques que ilustra un ejemplo de un codificador 308 y un decodificador 372. En este ejemplo, se ilustran el dispositivo electrónico A 302 y el dispositivo electrónico B 370. Sin embargo, cabe señalar que las características y la funcionalidad descritas en relación con el dispositivo electrónico A 302 y el dispositivo electrónico B 370 pueden combinarse en un solo dispositivo electrónico en algunas configuraciones.

El dispositivo electrónico A 302 incluye el codificador 308. El codificador 308 puede implementarse en hardware, software o una combinación de ambos. En una configuración, el codificador 308 puede ser un codificador de codificación de video de alta eficiencia (HEVC). Asimismo, se pueden usar otros codificadores. El dispositivo electrónico A 302 puede obtener una fuente 306. En algunas configuraciones, la fuente 306 puede capturarse en el dispositivo electrónico A 302 usando un sensor de imagen, recuperarse de la memoria o recibirse de otro dispositivo electrónico.

El codificador 308 puede codificar la fuente 306 para producir un flujo 334 de bits. Por ejemplo, el codificador 308 puede codificar una serie de imágenes (por ejemplo, video) en la fuente 306. El codificador 308 puede ser similar al codificador 108 descrito anteriormente en relación con la figura 1.

El flujo 334 de bits puede incluir datos de imagen codificados basados en la fuente 306. En algunas configuraciones, el flujo 334 de bits también puede incluir datos generales, como información de encabezado de segmento, información de PPS, etc. A medida que se codifican imágenes adicionales en la fuente 306, el flujo 334 de bits puede incluir una o más imágenes codificadas.

El flujo 334 de bits puede proporcionarse al decodificador 372. En un ejemplo, el flujo 334 de bits puede transmitirse al dispositivo electrónico B 370 usando un enlace alámbrico o inalámbrico. En algunos casos, esto se puede hacer a través de una red, como Internet o una red de área local (LAN). Como se ilustra en la figura 3, el decodificador 372 puede implementarse en el dispositivo electrónico B 370 por separado del codificador 308 en el dispositivo electrónico A 302. Sin embargo, cabe señalar que el codificador 308 y el decodificador 372 pueden implementarse en el mismo dispositivo electrónico en algunas configuraciones. En una implementación en la que el codificador 308 y el decodificador 372 se implementan en el mismo dispositivo electrónico, por ejemplo, el flujo 334 de bits puede proporcionarse a través de un bus al decodificador 372 o almacenarse en la memoria para su recuperación por el decodificador 372.

El decodificador 372 puede implementarse en hardware, software o una combinación de ambos. En una configuración, el decodificador 372 puede ser un decodificador de codificación de video de alta eficiencia (HEVC). Asimismo, se pueden usar otros decodificadores. El decodificador 372 puede ser similar al decodificador 272 descrito anteriormente en relación con la figura 2.

La figura 4 ilustra varios componentes que pueden utilizarse en un dispositivo electrónico 409. El dispositivo electrónico 409 puede implementarse como uno o más de los dispositivos electrónicos. Por ejemplo, el dispositivo electrónico 409 puede implementarse como el dispositivo electrónico 102 descrito anteriormente en conexión con la figura 1, como el dispositivo electrónico 270 descrito anteriormente en relación con la figura 2, o ambos.

El dispositivo electrónico 409 incluye un procesador 417 que controla el funcionamiento del dispositivo electrónico 409. El procesador 417 también puede denominarse CPU. La memoria 411, que puede incluir tanto memoria de solo lectura (ROM), memoria de acceso aleatorio (RAM) o cualquier tipo de dispositivo que pueda almacenar información, proporciona instrucciones 413a (por ejemplo, instrucciones ejecutables) y datos 415a al procesador 417. Una porción de la memoria 411 también puede incluir una memoria de acceso aleatorio no volátil (NVRAM). La memoria 411 puede estar en comunicación electrónica con el procesador 417.

Las instrucciones 413b y los datos 415b también pueden residir en el procesador 417. Las instrucciones 413b y/o los datos 415b cargados en el procesador 417 también pueden incluir instrucciones 413a y/o datos 415a de la memoria 411 que fueron cargados para su ejecución o procesamiento por el procesador 417. Las instrucciones 413b pueden ser ejecutadas por el procesador 417 para implementar una o más técnicas divulgadas en el presente documento.

El dispositivo electrónico 409 puede incluir una o más interfaces 419 de comunicación para comunicarse con otros dispositivos electrónicos. Las interfaces 419 de comunicación pueden estar basadas en tecnología de comunicación por cable, tecnología de comunicación inalámbrica o ambas. Los ejemplos de interfaces 419 de comunicación incluyen un puerto en serie, un puerto paralelo, un bus en serie universal (USB), un adaptador Ethernet, una interfaz de bus IEEE 1394, una interfaz de bus de interfaz de sistema informático pequeño (SCSI), un puerto de comunicación de infrarrojos (IR), un adaptador de comunicación inalámbrica Bluetooth, un transceptor inalámbrico de acuerdo con las especificaciones del proyecto de asociación de tercera generación (3GPP), etc.

El dispositivo electrónico 409 puede incluir uno o más dispositivos 423 de salida y uno o más dispositivos 421 de entrada. Ejemplos 423 de dispositivos de salida incluyen un altavoz, impresora, etc. Un tipo de dispositivo de salida que puede incluirse en un dispositivo electrónico 409 es un dispositivo 425 de visualización. Los dispositivos 425 de visualización usados con las configuraciones divulgadas en el presente documento pueden utilizar cualquier

tecnología de proyección de imagen adecuada, como un tubo de rayos catódicos (CRT), pantalla de cristal líquido (LCD), diodo de emisión de luz (LED), plasma de gas, electroluminiscencia o similares. Puede proporcionarse un controlador 427 de pantalla para convertir datos almacenados en la memoria 411 en texto, gráficos y/o imágenes en movimiento (según sea apropiado) mostrados en la pantalla 425. Ejemplos de dispositivos 421 de entrada incluyen un teclado, ratón, micrófono, dispositivo de control remoto, botón, mando, rueda de desplazamiento, panel táctil, pantalla táctil, lápiz óptico, etc.

Los diversos componentes del dispositivo electrónico 409 están acoplados entre sí mediante un sistema 429 de bus, que puede incluir un bus de potencia, un bus de señales de control y un bus de señales de estado, además de un bus de datos. Sin embargo, en aras de la claridad, los diversos buses se ilustran en la figura 4 como sistema 429 de bus. El dispositivo electrónico 409 ilustrado en la figura 4 es un diagrama de bloques funcional en lugar de una lista de componentes específicos.

El término "medio legible por computadora" se refiere a cualquier medio disponible al que se pueda acceder mediante una computadora o un procesador. El término "medio legible por computadora", como se usa en el presente documento, puede indicar un medio legible por computadora y/o procesador que no es transitorio y tangible. A modo de ejemplo, y no de limitación, un medio legible por computadora o legible por procesador puede comprender RAM, ROM, EEPROM, CD-ROM u otro almacenamiento en disco óptico, almacenamiento en disco magnético u otros dispositivos de almacenamiento magnético, o cualquier otro medio que se puede usar para transportar o almacenar el código informático deseado en forma de instrucciones o estructuras de datos y al que se puede acceder mediante una computadora o un procesador. El disco y el disco, como se usa en el presente documento, incluyen disco compacto (CD), disco láser, disco óptico, disco versátil digital (DVD), disquete y disco Blu-ray (marca registrada) donde los discos generalmente reproducen datos magnéticamente, mientras que los discos reproducen datos ópticamente con láseres. El código para el decodificador y/o codificador puede almacenarse en un medio legible por computadora.

Una imagen de entrada que comprende una pluralidad de bloques de árbol codificados (por ejemplo, generalmente denominados en el presente documento como bloques) puede dividirse en uno o varios segmentos. Los valores de las muestras en el área de la imagen que representa un segmento se pueden decodificar correctamente sin el uso de datos de otros segmentos, siempre que las imágenes de referencia usadas en el codificador y el decodificador sean las mismas y que el filtrado de desbloqueo no use información a través de los límites de segmento. Por lo tanto, la decodificación de entropía y la reconstrucción de bloques de un segmento no depende de otros segmentos. En particular, el estado de codificación de entropía se puede restablecer al comienzo de cada segmento. Los datos en otros segmentos pueden señalarse como no disponibles al definir la disponibilidad de cercanía para la decodificación y reconstrucción de entropía. Los segmentos pueden decodificarse por entropía y reconstruirse en paralelo. Preferiblemente, no se permite la intrapredicción y la predicción del vector de movimiento a través del límite de un segmento. Por el contrario, el filtrado de desbloqueo puede usar información a través de los límites de los segmentos.

La figura 5 ilustra una imagen 500 de vídeo de ejemplo que comprende once bloques en la dirección horizontal y nueve bloques en la dirección vertical (nueve bloques de ejemplo etiquetados como 501-509). La figura 5 ilustra tres segmentos de ejemplo: un primer segmento denominado "SEGMENTO # 0" 520, un segundo segmento denominado "SEGMENTO # 1" 530 y un tercer segmento denominado "SEGMENTO # 2" 540. El decodificador puede decodificar y reconstruir los tres segmentos 520, 530, 540, en paralelo. Cada uno de los segmentos puede transmitirse en orden de línea de exploración de manera secuencial. Al comienzo del proceso de decodificación/reconstrucción para cada segmento, los modelos de contexto se inicializan o reinician y los bloques en otros segmentos se señalan como no disponibles tanto para la decodificación de entropía como para la reconstrucción de bloques. El modelo de contexto generalmente representa el estado del codificador y/o decodificador de entropía. Por lo tanto, para un bloque, por ejemplo, el bloque etiquetado 503, en "SEGMENTO # 1", los bloques (por ejemplo, bloques etiquetados 501 y 502) en "SEGMENTO # 0" no pueden usarse para la selección o reconstrucción del modelo de contexto. Mientras que, para un bloque, por ejemplo, el bloque etiquetado 505, en "SEGMENTO # 1", otros bloques (por ejemplo, bloques etiquetados 503 y 504) en "SEGMENTO # 1" pueden usarse para la selección o reconstrucción del modelo de contexto. Por lo tanto, la decodificación de entropía y la reconstrucción de bloques se realizan en serie dentro de un segmento. A menos que los segmentos se definan mediante una ordenación de bloques flexible (FMO), los bloques dentro de un segmento se procesan en el orden de un escaneo de ráster.

La figura 6 representa una asignación de bloque de ejemplo en tres grupos de segmentos: un primer grupo de segmentos denominado "GRUPO DE SEGMENTOS # 0" 550, un segundo grupo de segmentos denominado "GRUPO DE SEGMENTOS # 1" 560 y un tercer grupo de segmentos denominado "GRUPO DE SEGMENTOS # 2" 570. Estos grupos 550, 560, 570 de segmentos pueden estar asociados con dos regiones de primer plano y una región de fondo, respectivamente, en la imagen 580.

La disposición de los segmentos, como se ilustra en la figura 5, puede limitarse a definir cada segmento entre un par de bloques en el orden de escaneo de la imagen, también conocido como escaneo de ráster u orden de escaneo de ráster. Esta disposición de segmentos de orden de escaneo es computacionalmente eficiente pero no tiende a prestarse a la codificación y decodificación paralelas altamente eficientes. Además, esta definición del orden de

escaneo de los segmentos tampoco tiende a agrupar regiones localizadas más pequeñas de la imagen que probablemente tengan características comunes muy adecuadas para la eficiencia de la codificación. La disposición de los segmentos, como se ilustra en la figura 6, es muy flexible en su disposición, pero no tiende a prestarse a una codificación o decodificación paralela de alta eficiencia. Además, esta definición altamente flexible de segmentos es computacionalmente compleja de implementar en un decodificador.

Con referencia a la figura 7, una técnica de mosaico divide una imagen en un conjunto de regiones rectangulares (inclusive cuadradas). Los bloques (también denominados unidades de codificación más grandes o bloques de árbol codificados en algunos sistemas) dentro de cada uno de los mosaicos se codifican y decodifican en un orden de escaneo de ráster. La disposición de los mosaicos también se codifica y decodifica en un orden de escaneo de ráster. Por consiguiente, puede haber cualquier número adecuado de límites de columna (por ejemplo, 0 o más) y puede haber cualquier número adecuado de límites de fila (por ejemplo, 0 o más). Por tanto, la trama puede definir uno o más segmentos, como el segmento que se ilustra en la figura 7. En algunas realizaciones, los bloques ubicados en diferentes mosaicos no están disponibles para intrapredicción, compensación de movimiento, selección de contexto de codificación de entropía u otros procesos que se basan en información de bloques cercanos.

Con referencia a la figura 8, se muestra la técnica del mosaico dividiendo una imagen en un conjunto de tres columnas rectangulares. Los bloques (también denominados unidades de codificación más grandes o bloques de árbol codificados en algunos sistemas) dentro de cada uno de los mosaicos se codifican y decodifican en un orden de escaneo de ráster. Los mosaicos también se codifican y decodifican en un orden de escaneo de ráster. Se pueden definir uno o más segmentos en el orden de escaneo de los mosaicos. Cada uno de los segmentos se puede decodificar de forma independiente. Por ejemplo, el segmento 1 puede definirse como que incluye los bloques 1-9, el segmento 2 puede definirse como que incluye los bloques 10-28 y el segmento 3 puede definirse como que incluye los bloques 29-126 que abarcan tres mosaicos. El uso de mosaicos facilita la eficiencia de la codificación al procesar datos en regiones más localizadas de una trama.

Cabe señalar que, en algunos casos, la codificación de video puede no incluir opcionalmente mosaicos, y puede incluir opcionalmente el uso de un patrón de codificación/decodificación de frente de onda para las tramas del video. De esta manera, una o más líneas del video (como una pluralidad de grupos de una o más filas de macrobloques (o bloques de árbol codificados alternativamente), siendo cada uno de los grupos representativo de un subflujo de frente de onda, puede codificarse/decodificarse de forma paralela. En general, la división del video se puede construir de cualquier manera adecuada.

Los estándares de codificación de video a menudo comprimen los datos de video para su transmisión a través de un canal con ancho de banda de frecuencia limitado y/o capacidad de almacenamiento limitada. Estos estándares de codificación de video pueden incluir múltiples etapas de codificación como intrapredicción, transformación de dominio espacial a dominio frecuencia, cuantificación, codificación de entropía, estimación de movimiento y compensación de movimiento, para codificar y decodificar tramas de manera más efectiva. Muchas de las etapas de codificación y decodificación son excesivamente complejas computacionalmente.

Se han desarrollado varias técnicas de codificación de video escalables. En la codificación de vídeo escalable, un decodificador recibe un flujo de bits primario (generalmente denominado flujo de bits de capa base). Además, el decodificador puede recibir uno o más flujos de bits secundarios (generalmente denominados capa o capas de mejora). La función de cada capa de mejora puede ser: mejorar la calidad del flujo de bits de la capa base; para mejorar la velocidad de tramas del flujo de bits de la capa base; y/o para mejorar la resolución de píxeles del flujo de bits de la capa base. La escalabilidad de calidad también se conoce como escalabilidad de relación señal/ruido (SNR). La escalabilidad de la velocidad de tramas también se conoce como escalabilidad temporal. La escalabilidad de resolución también se conoce como escalabilidad espacial.

La capa o capas de mejora pueden cambiar otras características del flujo de bits de la capa base. Por ejemplo, una capa de mejora se puede asociar con una relación de aspecto y/o ángulo de visión diferente a la capa base. Otro aspecto de las capas de mejora es que la capa base y una capa de mejora pueden corresponder a diferentes estándares de codificación de video, por ejemplo, la capa base puede ser MPEG-2 (grupo 2 de expertos en imágenes en movimiento) y una capa de mejora puede ser HEVC-Ext (extensión de codificación de video de alta eficiencia).

Se puede definir un orden entre capas. Por ejemplo:

Capa base (más baja) [capa 0]

Capa 0 de mejora [capa 1]

Capa 1 de mejora [capa 2]

Capa n de mejora (más alta) [capa n + 1]

La capa o capas de mejora pueden depender entre sí (además de la capa base). En un ejemplo, la capa 2 de mejora sólo se puede usar si al menos una porción de la capa 1 de mejora se ha analizado y/o reconstruido con éxito (y si al menos una porción de la capa base se ha analizado y/o reconstruido con éxito).

5 El flujo de bits del vídeo codificado puede incluir una estructura de sintaxis que se coloca en paquetes de datos lógicos generalmente denominados unidades de capa de abstracción de red (NAL). Cada unidad NAL incluye un encabezado de unidad NAL, tal como un encabezado de unidad NAL de dos bytes (por ejemplo, 16 bits), para identificar el propósito de la carga útil de datos asociada. Por ejemplo, cada segmento codificado (y/o imagen) puede codificarse en una o más unidades NAL de segmento (y/o imagen). Se pueden incluir otras unidades NAL para otras
10 categorías de datos, como por ejemplo, información de mejora suplementaria, segmento codificado de imagen de acceso a subcapa temporal (TSA), segmento codificado de imagen de acceso a subcapa temporal escalonado (STSA), segmento codificado de imagen final sin TSA ni STSA, segmento codificado de imagen de acceso a enlace roto, segmento codificado de imagen de actualización decodificada instantánea, segmento codificado de imagen de acceso aleatorio limpio, segmento codificado de imagen inicial decodificable, segmento codificado de etiquetado
15 para descartar imagen, conjunto de parámetros de vídeo, conjunto de parámetros de secuencia, conjunto de parámetros de imagen, delimitador de unidad de acceso, final de secuencia, final de flujo de bits, datos de relleno y/o mensaje de información de mejora de secuencia. Se pueden incluir otros tipos de unidades NAL, según se desee.

20 Una imagen de imagen de punto de acceso aleatorio (RAP) contiene solo segmentos I y puede ser una imagen de acceso de enlace roto (BLA), una imagen de acceso aleatorio limpio (CRA) o una imagen de actualización de decodificación instantánea (IDR). La primera imagen del flujo de bits es una imagen RAP.

25 Una imagen de imagen de acceso a enlace roto (BLA) es un tipo de imagen RAP. Una imagen BLA contiene solo segmentos I y puede ser la primera imagen en el flujo de bits en el orden de decodificación, o puede aparecer más tarde en el flujo de bits. Cada imagen BLA comienza una nueva secuencia de vídeo codificada y tiene el mismo efecto en el proceso de decodificación que una imagen IDR. Sin embargo, una imagen BLA contiene elementos de sintaxis que, si hubiera sido una imagen CRA en su lugar, especificarían un conjunto de imágenes de referencia no vacías. Cuando se encuentra una imagen BLA en un flujo de bits, estos elementos de sintaxis se ignoran y el conjunto de imágenes de referencia se inicializa en su lugar como un conjunto vacío.

30 Una imagen de acceso aleatorio limpio (CRA) es un tipo de imagen RAP. Una imagen CRA contiene solo segmentos I y puede ser la primera imagen en el flujo de bits en el orden de decodificación, o puede aparecer más tarde en el flujo de bits. Una imagen CRA puede tener imágenes iniciales decodificables asociadas (DLP) e imágenes etiquetadas para descartar (TFD).

35 Una imagen de actualización de decodificación instantánea (IDR) es un tipo de imagen RAP. Una imagen IDR contiene solo segmentos I y puede ser la primera imagen en el flujo de bits en el orden de decodificación, o puede aparecer más adelante en el flujo de bits. Cada imagen IDR es la primera imagen de una secuencia de vídeo codificada en orden de decodificación.

40 Una imagen inicial decodificable (DLP) son imágenes iniciales. Las imágenes DLP no se usan como imágenes de referencia para el proceso de decodificación de imágenes finales de la misma imagen RAP asociada.

45 Una imagen etiquetada para descartar (TFD) son imágenes iniciales de una imagen BLA o CRA asociada. Cuando la imagen RAP asociada es una imagen BLA o es la primera imagen codificada en el flujo de bits, la imagen TFD no se emite y puede que no se pueda decodificar correctamente, ya que la imagen TFD puede contener referencias a imágenes de referencia que no están presentes en el flujo de bits.

50 Una imagen inicial es una imagen que precede a la imagen RAP asociada en el orden de salida.

Una imagen final es una imagen que sigue a la imagen RAP asociada en el orden de salida.

55 La unidad NAL proporciona la capacidad de mapear los datos de la capa de codificación de vídeo (VCL) que representan el contenido de las imágenes en varias capas de transporte. Las unidades NAL pueden clasificarse en unidades NAL VCL y no VCL según contengan imagen codificada u otros datos asociados, respectivamente. B. Bros, W.-J. Han, J.-R. Ohm, G.J. Sullivan y T.-Wiegand, "Versión 8 de la especificación de texto de codificación de vídeo de alta eficiencia (HEVC)", JCTVC-J10003, Estocolmo, julio de 2012; "BoG sobre sintaxis de alto nivel para la planificación de extensiones", Ye-Kui Wang, JCTVC-J00574, julio de 2012; y "BoG sobre sintaxis de alto nivel para la planificación de extensiones", Ye-Kui Wang, JCTVC-J00574r1, julio de 2012, por la presente se consideran técnica
60 anterior relevante.

65 Con referencia a la figura 9A, la sintaxis del encabezado de la unidad NAL puede incluir dos bytes de datos, a saber, 16 bits. El primer bit es un "forbidden_zero_bit" que siempre se pone a cero al comienzo de una unidad NAL. Los siguientes seis bits son un "nal_unit_type" que especifica el tipo de estructura de datos de cargas útiles de secuencia de bytes sin procesar ("RBSP") contenida en la unidad NAL. Los siguientes 6 bits son "nuh_reserved_zero_6bits". Nuh_reserved_zero_6bits puede ser igual a 0 en la especificación base del estándar. Se pueden especificar otros

valores de `nuh_reserved_zero_6bits` según se desee. Los decodificadores pueden ignorar (es decir, eliminar del flujo de bits y descartar) todas las unidades NAL con valores de `nuh_reserved_zero_6bits` no iguales a 0 cuando se maneja un flujo basado en la especificación básica del estándar. En una extensión escalable o de otro tipo, `nuh_reserved_zero_6bits` puede especificar otros valores, para señalar la codificación de video escalable y/o extensiones de sintaxis. En algunos casos, el elemento de sintaxis `nuh_reserved_zero_6bits` puede llamarse `reserved_zero_6bits`. En algunos casos, el elemento de sintaxis `nuh_reserved_zero_6bits` se puede llamar como `layer_id_plus1` o `layer_id`, como se ilustra en la figura 9B y la figura 9C. En este caso, el elemento `layer_id` será `layer_id_plus1` menos 1. En este caso, puede usarse para señalar información relacionada con la capa de video codificado escalable. El siguiente elemento de sintaxis es "`nuh_temporal_id_plus1`". `nuh_temporal_id_plus1` menos 1 puede especificar un identificador temporal para la unidad NAL. El identificador temporal variable `TemporalId` puede especificarse como `TemporalId = nuh_temporal_id_plus1 - 1`.

Con referencia a la figura 10, se ilustra una estructura de sintaxis de unidad NAL general. La sintaxis de dos bytes del encabezado de la unidad NAL de la figura 9 se incluye en la referencia a `nal_unit_header()` de la figura 10. El resto de la sintaxis de la unidad NAL se relaciona principalmente con la RBSP.

Una técnica existente para usar "`nuh_reserved_zero_6bits`" es señalar información de codificación de video escalable dividiendo los 6 bits de `nuh_reserved_zero_6bits` en distintos campos de bits, a saber, uno o más de un ID de dependencia, un ID de calidad, un ID de vista y una marca de profundidad, cada uno de los cuales se refiere a la identificación de una capa diferente del video codificado escalable. En consecuencia, los 6 bits indican a qué capa de la técnica de codificación escalable pertenece esta unidad NAL en particular. Luego, en una carga útil de datos, tal como una sintaxis de extensión ("`scalability_type`") de conjunto de parámetros de video ("`VPS`") como se ilustra en la figura 11, se define la información sobre la capa. La sintaxis de extensión de VPS de la figura 11 incluye 4 bits para el tipo de escalabilidad (elemento de sintaxis `scalability_type`) que especifica los tipos de escalabilidad en uso en la secuencia de video codificada y las dimensiones señalizadas a través de `layer_id_plus1` (o `layer_id`) en el encabezado de la unidad NAL. Cuando el tipo de escalabilidad es igual a 0, la secuencia de video codificada se ajusta a la especificación base, por lo que `layer_id_plus1` de todas las unidades NAL es igual a 0 y no hay unidades NAL que pertenezcan a una capa o vista de mejora. Los valores más altos del tipo de escalabilidad se interpretan como se ilustra en la figura 12.

`Layer_id_dim_len[i]` especifica la longitud, en bits, del *i*-ésimo ID de dimensión de escalabilidad. La suma de los valores `layer_id_dim_len[i]` para todos los valores *i* en el rango de 0 a 7 es menor o igual que 6. `vps_extension_byte_alignment_reserved_zero_bit` es cero. `vps_layer_id[i]` especifica el valor de `layer_id` de la *i*-ésima capa a la que se aplica la siguiente información de dependencia de la capa. `num_direct_ref_layers[i]` especifica el número de capas de las que depende directamente la *i*-ésima capa. `ref_layer_id[i][j]` identifica la *j*-ésima capa de la que depende directamente la *i*-ésima capa.

De esta manera, la técnica existente señala los identificadores de escalabilidad en la unidad NAL y en el conjunto de parámetros de video para asignar los bits entre los tipos de escalabilidad enumerados en la figura 12. Luego, para cada tipo de escalabilidad, la figura 12 define cuántas dimensiones se soportan. Por ejemplo, el tipo de escalabilidad 1 tiene 2 dimensiones (es decir, espacial y de calidad). Para cada una de las dimensiones, `layer_id_dim_len[i]` define el número de bits asignados a cada una de estas dos dimensiones, donde la suma total de todos los valores de `layer_id_dim_len[i]` es menor o igual a 6, que es el número de bits en `nuh_reserved_zero_6bits` del encabezado de la unidad NAL. Por tanto, en combinación, la técnica identifica qué tipos de escalabilidad están en uso y cómo se asignan los 6 bits del encabezado de la unidad NAL entre la escalabilidad.

Aunque tal combinación fija de diferentes dimensiones de escalabilidad, como se ilustra en la figura 12, es adecuado para muchas aplicaciones, existen combinaciones deseables que no están incluidas. Con referencia a la figura 13, una sintaxis de extensión de conjunto de parámetros de video modificada especifica un tipo de escalabilidad para cada bit en el elemento de sintaxis `nuh_reserved_zero_6bits`. `vps_extension_byte_alignment_reserved_zero_bit` se establece en 0. `Max_num_layers_minus1_bits` indica el número total de bits usados para el elemento de sintaxis en los dos primeros bytes del encabezado de la unidad NAL en la figura 9 denominado `layer_id_plus1` o `nuh_reserved_zero_6bits`. `scalability_map[i]` especifica el tipo de escalabilidad para cada bit en el elemento de sintaxis `layer_id_plus1`. En algunos casos, el elemento de sintaxis `layer_id_plus1` puede llamarse en su lugar `nuh_reserved_zero_6bits` o elemento de sintaxis `rserved_zero_6bits`. El mapa de escalabilidad para todos los bits del elemento de sintaxis `layer_id_plus1` juntos especifica la escalabilidad en uso en la secuencia de video codificada. El valor real del identificador para cada uno de los tipos de escalabilidad se señala a través de los bits correspondientes en el campo `layer_id_plus1` (`nuh_reserved_zero_6bits`) en el encabezado de la unidad NAL. Cuando `scalability_map[i]` es igual a 0 para todos los valores de *i*, la secuencia de video codificada se ajusta a la especificación base, por lo que el valor `layer_id_plus1` de las unidades NAL es igual a 0 y no hay unidades NAL que pertenezcan a una capa o vista de mejora. `vps_layer_id[i]` especifica el valor de `layer_id` de la *i*-ésima capa a la que se aplica la siguiente información de dependencia de la capa. `Num_direct_ref_layers[i]` especifica el número de capas de las que depende directamente la *i*-ésima capa. El `ref_layer_id[i][j]` identifica la *j*-ésima capa de la que depende directamente la *i*-ésima capa.

Los valores más altos del mapa de escalabilidad [i] se interpretan como se muestra en la figura 14. El mapa de escalabilidad [i] incluye las dimensiones de escalabilidad de (0) ninguna; (1) espacial; (2) calidad; (3) profundidad; (4) vista múltiple; (5) sin especificar; (6) reservado; y (7) reservado.

5 Por lo tanto, cada bit en el encabezado de la unidad NAL se interpreta basándose en los 3 bits en el conjunto de parámetros de video de cuál es la dimensión de escalabilidad (por ejemplo, ninguno, espacial, calidad, profundidad, vista múltiple, sin especificar, reservado). Por ejemplo, para señalar que todos los bits en layer_id_plus1 corresponden a la escalabilidad espacial, los valores del mapa de escalabilidad en el VPS pueden codificarse como 001 001 001 001 001 001 para los 6 bits del encabezado de la unidad NAL. También, por ejemplo, para señalar que 3 bits en layer_id_plus1 corresponden a escalabilidad espacial y 3 bits corresponden a escalabilidad de calidad, los valores del mapa de escalabilidad en el VPS pueden codificarse como 001 001 001 010010010 para los 6 bits del encabezado de unidad NAL.

15 Con referencia a la figura 15, otro ejemplo presente con fines ilustrativos solo incluye el conjunto de parámetros de video que indica el número de dimensiones de escalabilidad en los 6 bits del encabezado de la unidad NAL usando num_scalability_dimensions_minus1. Num_scalability_dimensions_minus1 más 1 indica el número de dimensiones de escalabilidad señalizadas a través de los elementos de sintaxis layer_id_plus1; nuh_reserved_zero_6bits; y/o reserved_zero_6bits. El mapa de escalabilidad [i] tiene la misma semántica que la descrita anteriormente en relación con la figura 13. El num_bits_for_scalability[i] map especifica la longitud en bits para la i-ésima dimensión de escalabilidad. La suma de todos los num_bits_for_scalability_map[i] para i=0,... num_scalability_dimensions_minus1 es igual a seis (o igual al número de bits usados para los elementos de sintaxis layer_id_plus1; vps_reserved_zero_6bits; maxnum_layers_minus1; reserved_zero_x6bits; nuh_reserved_zero_6bits).

25 Con respecto a la figura 13 y la figura 15 se pueden usar otras variaciones, si se desea. Por ejemplo, scalability_map[i] se puede señalar con u(4) (o u(n) con n>3 o n<3). En este caso, los valores más altos del mapa de escalabilidad [i] pueden especificarse como reservados para flujos de bits que se ajusten a un perfil particular de la técnica de video. Por ejemplo, los valores de mapa de escalabilidad 6... 15 pueden especificarse como 'reservados' cuando se señala scalability_map[i] con u(4). Por ejemplo, el mapa de escalabilidad [i] puede indicarse con ue(v) o algún otro esquema de codificación. Por ejemplo, se puede especificar una restricción de modo que los valores del scalability_map[i] estén dispuestos en un orden monótono no decreciente (o no creciente). Esto da como resultado que varios campos de dimensión de escalabilidad en el campo layer_id_plus1 en el encabezado de la unidad NAL sean contiguos.

35 Otra técnica existente para señalar la codificación de video escalable usando el elemento de sintaxis "layer_id_plus1" o "nuh_reserved_zero_6bits" es mapear el layer_id_plus1 en el encabezado de la unidad NAL a una identificación de capa señalizando una tabla de búsqueda general en el conjunto de parámetros de video. Con referencia a la figura 16, la técnica existente incluye un conjunto de parámetros de video que especifica el número de tipos de dimensión e identificaciones de dimensión para la i-ésima capa de la tabla de búsqueda. En particular, vps_extension_byte_alignment_reserved_zero_bit es cero. Num_dimensions_minus1[i] más 1 especifica el número de tipos de dimensión (dimension_type[i][j]) e identificadores de dimensión (dimension_id[i][j]) para la i-ésima capa. El dimension_type[i][j] especifica el j-ésimo tipo de dimensión de escalabilidad de la i-ésima capa, que tiene layer_id o layer_id_plus1 igual a i, como se especifica en la figura 17. Como se ilustra en la figura 17, las dimensiones que se identifican incluyen (0) orden de vista idx; (1) marca de profundidad; (2) ID de dependencia; (3) ID de calidad; (4)-(15) reservado. El dimension_id[i][j] especifica el identificador del j-ésimo tipo de dimensión de escalabilidad de la i-ésima capa, que cuando no está presente se infiere que es 0. num_direct_ref_layers[i] especifica el número de capas de las que depende directamente la i-ésima capa. El ref_layer_id[i][j] identifica la j-ésima capa de la que depende directamente la i-ésima capa. Desafortunadamente, el ejemplo presente con fines ilustrativos solo ilustrado en la figura 16 da como resultado una tabla de búsqueda grande y difícil de manejar.

50 Con referencia a la figura 18, una extensión de conjunto de parámetros de video modificado incluye una máscara de escalabilidad que se usa en combinación con una dimensión de escalabilidad. La máscara de escalabilidad señala un patrón de 0 y 1 bits con cada bit correspondiente a una dimensión de escalabilidad como se indica por la sintaxis del mapa de escalabilidad de la figura 19 o 29. Un valor de 1 para una dimensión de escalabilidad particular indica que esta dimensión de escalabilidad está presente en esta capa (i-ésima capa). Un valor de 0 para una dimensión de escalabilidad particular indica que esta dimensión de escalabilidad no está presente en esta capa (capa i-ésima). Por ejemplo, un conjunto de bits de 00100000 se refiere a la escalabilidad de calidad. El valor de identificador real de la dimensión de escalabilidad particular que está presente se indica mediante el valor de scalability_id[j] señalado. Los valores de num_scalability_types[i] es igual a la suma del número de bits en la máscara de escalabilidad que

$$\text{num_scalability_types}[i] = \sum_{k=0}^7 \text{scalability_mask}[i](k)$$

60 tiene el valor 1. Por tanto, El scalability_id[j] indica el valor identificador de la dimensión j-ésima de escalabilidad para el tipo de valores de escalabilidad que están señalizados por el valor de la máscara de escalabilidad.

Con referencia a la figura 20, una modificación de la figura 18, incluye la máscara de escalabilidad señalizada fuera del bucle. Esto da como resultado una máscara común para cada identificación de capa. Con referencia a la figura 21, en esta modificación, un conjunto de parámetros de vídeo de ejemplo correspondiente puede incluir la identificación escalable sin que se incluya la máscara de escalabilidad. En este caso, el elemento sintáctico scalable
 5 `_id[i]` tiene la misma interpretación que el elemento sintáctico scalability `_id[i]` en la figura 18.

Con referencia a la figura 22, una modificación de la figura 18 incluye indicadores de información de fuentes adicionales que se están señalizando. La información del tipo de escaneo de origen se señala por
 10 `source_scan_type_info_idc[i]` para cada capa *i*. `source_scan_type_info_idc[i]` igual a 0 indica que la capa *i* tiene imágenes cuyo tipo de escaneo debe interpretarse como tipo de escaneo entrelazado para todas las imágenes. `source_scan_type_info_idc[i]` igual a 1 indica que la capa *i* tiene imágenes cuyo tipo de escaneo debe interpretarse como tipo de escaneo progresivo para todas las imágenes. `source_scan_type_info_idc[i]` igual a 2 indica que la capa *i* tiene imágenes cuyo tipo de escaneo es desconocido. `source_scan_type_info_idc[i]` igual a 3 indica que la capa *i* tiene imágenes que son una mezcla de tipo de escaneo progresivo y tipo de escaneo entrelazado. Esto se indica en
 15 la figura 23.

La información de origen 2D/3D se señala por `source_2d_3d_info_idc[i]` para cada capa *i*. `source_2d_3d_info_idc[i]` igual a 1 indica que todas las imágenes de la capa *i* están empaquetadas en un formato 3D compatible con tramas. `source_2d_3d_info_idc[i]` igual a 0 indica que todas las imágenes de la capa *i* son tramas 2D, es decir, ninguna de
 20 ellas está empaquetada en un formato 3D compatible con tramas. `source_2d_3d_info_idc[i]` igual a 2 indica que la capa *i* tiene imágenes cuya disposición de empaquetado de tramas se desconoce. `source_2d_3d_info_idc[i]` igual a 3 indica que la capa *i* tiene imágenes que son una mezcla de tramas 3D y 2D compatibles con tramas. Esto se indica en la figura 24.

En otra variante, los elementos de sintaxis `source_scan_type_info_idc[i]` y/o `source_2d_3d_info_idc[i]` pueden indicarse con `u(1)` o `u(n)` con $n > 3$. O puede indicarse con `ue(v)` o algún otro método de codificación.

En otro ejemplo presente con fines ilustrativos, sólo se pueden usar nombres diferentes para los elementos de sintaxis que los nombres `source_scan_info_idc[i]` y `source_2d_3d_info_idc[i]`.
 30

En otro ejemplo presente con fines ilustrativos, sólo los elementos de sintaxis `source_scan_type_info_idc[i]` y/o `source_2d_3d_info_idc[i]` pueden señalizarse en el conjunto de parámetros de secuencia (SPS)/conjunto de parámetros de imagen (PPS)/encabezado de segmento u otra parte normativa del flujo de bits.

En otro ejemplo presente con fines ilustrativos, solo los elementos de sintaxis `source_scan_type_info_idc` y `source_2d_3d_info_idc` pueden señalizarse solo una vez si son iguales para todas las capas *[i]*. En este caso, los elementos de sintaxis se señalizan una vez como `source_scan_type_info_idc` y `source_2d_3d_info_idc` sin el índice de capa *i*. Esto se muestra en la figura. 25.
 35

En otro ejemplo presente con fines ilustrativos, sólo los elementos de sintaxis `source_scan_type_info_idc` y/o `source_2d_3d_info_idc` pueden señalizarse también para la capa base. Esto se muestra en la figura 26.
 40

En otro ejemplo presente con fines ilustrativos, sólo los elementos de sintaxis `source_scan_type_info_idc [i]` y `source_2d_3d_info_idc` pueden señalizarse en otro lugar en el VPS diferente al mostrado en la figura 22; la figura 25, la figura 26. Puede estar señalizado en el VPS base y/o en `vps_extension()`.
 45

En un ejemplo presente con fines ilustrativos, sólo los elementos de sintaxis `source_scan_type_info_idc [i]` y/o `source_2d_3d_info_idc [i]` pueden indicarse con `u(3)` en lugar de `u(2)`. Los valores para ellos se muestran respectivamente en la figura 27 y la figura 28.
 50

También en la figura 22 y la figura 25, `avc_base_codec_flag` igual a 1 especifica que la capa base se ajusta a la Rec. UIT-T H.264 | ISO/IEC 14496-10 y `avc_base_codec_flag` igual a 0 especifica HEVC. `vps_nuh_layer_id_present_flag` indica si se señala el ID de capa en `nuh[i]` variable que señala el valor del ID de capa en el encabezado de la unidad NAL.
 55

En una realización, uno o más de los elementos de sintaxis `scalability_mask[i]`, máscara de escalabilidad, escalabilidad `id[i]` pueden señalizarse usando un número de bits diferente al de `u(8)`. Por ejemplo, se podrían señalar con `u(16)` (o `u(n)` con $n > 8$ o $n < 8$). En otra realización, uno o más de estos elementos de sintaxis podrían señalizarse con `ue(v)`. En otra realización, la `scalability_mask` se puede señalar en el encabezado de la unidad NAL en los elementos de sintaxis `layer_id_plus1`; `vps_reserved_zero_6bits`; `max_num_layers_minus1`; `reserved_zero_6bits`; y/o `nuh_reserved_zero_6bits`. En algunas realizaciones, el sistema puede hacer esto solo para unidades NAL de VPS, o solo para unidades que no son NAL de VPS, o para todas las unidades NAL. En otra realización más, la `scalability_mask` se puede señalar por imagen en cualquier lugar del flujo de bits. Por ejemplo, puede señalizarse en el encabezado de segmento, el conjunto de parámetros de imagen, el conjunto de parámetros de vídeo o cualquier otro conjunto de parámetros o cualquier otra parte normativa del flujo de bits.
 60
 65

Cabe señalar que las figuras 13, 15, 18, 20, 21, 22 y la descripción correspondiente se refieren a 6 bits, ya que el elemento de sintaxis `nuh_reserved_zero_6bits` o `layer_id_plus1` en el encabezado de la unidad NAL de la figura 9 tiene 6 bits. Sin embargo, toda la descripción anterior puede modificarse adecuadamente si ese elemento de sintaxis usa un número de bits diferente de 6 bits. Por ejemplo, si ese elemento de sintaxis (`nuh_reserved_zero_6bits` o `layer_id_plus1`) en su lugar usó 9 bits, entonces en la figura 13 el valor de `max_num_layer_minus1` bits será 9 y el `scalability_map[i]` se señalará para cada uno de los 9 bits en lugar de 6 bits.

Con referencia a la figura 30 una modificación de la figura 18 incluye la máscara de escalabilidad (`scalability_mask`) que se señala fuera del bucle. Esto da como resultado una máscara común para cada identificación de capa. `Scalability_mask` señala un patrón de 0 y 1 bits con cada bit correspondiente a una dimensión de escalabilidad como se indica por la sintaxis del mapa de escalabilidad de la figura 31. Un valor de 1 para una dimensión de escalabilidad particular indica que esta dimensión de escalabilidad está presente en esta capa (i-ésima capa). Un valor de 0 para una dimensión de escalabilidad particular indica que esta dimensión de escalabilidad no está presente en esta capa (i-ésima capa). Por ejemplo, un conjunto de bits de 00100000 se refiere a la escalabilidad de calidad. El valor de identificador real de la dimensión de escalabilidad particular que está presente se indica mediante el valor de `scalability_id[j]` señalado. Los valores de `num_scalability_types[i]` es igual a la suma del número de bits en la `scalability_mask` que tiene el valor 1. Por tanto,

$$\text{NumScalabilityTypes}[i] = \sum_{k=0}^{15} \text{scalability_mask}(k)$$

En este caso, la variable `scalability_id[j]` puede denominarse `dimension_id[i][j]` variable. `dimension_id[i][j]` especifica el identificador de escalabilidad de la j-ésima dimensión de escalabilidad de la i-ésima capa. Entonces, una variable `ScalabilityId[i][j]` se deriva de la siguiente manera.

```
for(i = 1; i <= vps_max_layers_minus1; i++) {
    for(k=0, j=0; k<=15; k++) {
        if(scalability_mask(k)==1)
            ScalabilityId [i] [k]=dimension_id[i] [j++]
        else
            ScalabilityId [i] [k]=0;
    }
}
```

Donde `ScalabilityId[i][k]` señala el ID de dimensión para el tipo de escalabilidad correspondiente de la siguiente manera.

k	ScalabilityId[i] [k]
0	Dependency[i] [k]
1	Quality[i][k]
2	depthFlag[i] [k]
3	ViewId[i] [k]
4-15	Reservado

Donde `DependencyId[i] [1]` es el ID de dependencia para la dimensión de escalabilidad espacial para la i-ésima capa, `QualityId[i] [2]` es el ID de calidad para la dimensión de escalabilidad de calidad para la i-ésima capa, `depthFlag [i] [3]` es el indicador de profundidad/ID de profundidad para la dimensión de escalabilidad de profundidad para la i-ésima capa, y `ViewId[i] [4]` es la ID de vista para la dimensión de escalabilidad multivista para la i-ésima capa.

También en la figura 30 `avc_base_codec_flag` igual a 1 especifica que la capa base se ajusta a la Rec. UIT-T H.264 | ISO/IEC 14496-10 y `avc_base_codec_flag` igual a 0 especifica HEVC. `vps_nuh_layer_id_presnet_flag` indica si se señala la variable `layer_id_in_nuh[i]` que señala el valor de `layer_id` en el encabezado de la unidad NAL.

En otra realización, uno o más de los elementos de sintaxis `scalability_mask[i]`, `scalability_mask`, `scalability_id[j]` pueden señalarse usando un número de bits diferente al de `u(8)`. Por ejemplo, se podrían señalar con `u(16)` (o `u(n)` con $n > 8$ o $n < 8$). En otra realización, uno o más de estos elementos de sintaxis podrían señalarse con `ue(v)`. En otra realización, la `scalability_mask` se puede señalar en el encabezado de la unidad NAL en los elementos de sintaxis `layer_id_plus1`; `vps_reserved_zero_6bits`; `max_num_layers_minus1`; `reserved_zero_6bits`; y/o

nuh_reserved_zero_6bits. En algunas realizaciones, el sistema puede hacer esto solo para unidades NAL de VPS, o solo para unidades que no son NAL de VPS, o para todas las unidades NAL. En otra realización más, la scalability_mask se puede señalar por imagen en cualquier lugar del flujo de bits. Por ejemplo, puede señalizarse en el encabezado de segmento, el conjunto de parámetros de imagen, el conjunto de parámetros de vídeo o cualquier otro conjunto de parámetros o cualquier otra parte normativa del flujo de bits.

Cabe señalar que las figuras 13, 15, 18, 20, 21, 30, 31 y la descripción correspondiente se refieren a 6 bits, ya que el elemento de sintaxis nuh_reserved_zero_6bits o layer_id_plus1 en el encabezado de la unidad NAL de la figura 9 tiene 6 bits. Sin embargo, toda la descripción anterior puede modificarse adecuadamente si ese elemento de sintaxis usa un número de bits diferente de 6 bits. Por ejemplo, si ese elemento de sintaxis (nuh_reserved_zero_6bits o layer_id_plus1) en su lugar usó 9 bits, entonces en la figura 13 el valor de max_num_layer_minus1 bits será 9 y el scalability_map[i] se señalará para cada uno de los 9 bits en lugar de 6 bits.

Con referencia a la figura 32 una modificación de la figura 30 incluye una marca adicional (vps_implicit_dimension_ids_flag) que indica la señalización implícita de identificadores de dimensión de escalabilidad y nuevos elementos de sintaxis (max_dimension_id_minus1[i]) que se señalizan. vps_implicit_dimension_ids_flag con un valor de 1 especifica que la lista explícita de valores de dimension_id[i][j] para varias dimensiones de escalabilidad no se señala y se infieren los valores de dimension_id[i][j]. vps_implicit_dimension_ids_flag con un valor de 0 especifica que se señala la lista explícita de los valores de dimension_id[i][j] para varias dimensiones de escalabilidad. max_dimension_id_minus1[i] especifica el valor máximo de dimension_id para la dimensión de escalabilidad i. Las dimensiones de escalabilidad se enumeran en la tabla scalability_mask, por ejemplo, en la figura 31.

Cuando los valores de dimension_id se señalizan implícitamente, se infieren como DimID[i][j] como se describe en el código siguiente durante el proceso de decodificación.

```
int Id[]={0,0,0,...,0};
for(i=0;i<=vps_max_layers_minus1;i++)
{
    for(j=0;j<NumScalabilityTypes;j++)
    {
        DimId[i][j]= Id[j];
    }
    Id[NumScalabilityTypes-1]= Id[NumScalabilityTypes-1]+1;
    for(k= NumScalabilityTypes-1;k>=0;k--)
    {
        if(Id[k]=(max_dimension_id_minus1[k]+1))
        {
            Id[k]=0;
            Id[k-1]= Id[k-1]+1;
        }
    }
}
```

Cuando se usa la señalización implícita de identificadores de dimensión de escalabilidad, se infiere que el valor de la identificación de la capa NUH es igual al índice de entrada (i) del bucle para (i = 0; i <= vps_max_layers_minus1; i++) en la figura 32.

La figura 34 muestra una sintaxis variante con funcionalidad similar a la sintaxis de la figura 32. Todos los elementos de sintaxis de la figura 34 tienen el mismo significado semántico que en la figura 32. En la figura 34 algunos de los elementos de sintaxis (vps_implicit_dimension_ids_flag, max_dimension_id_minus1[i]) se mueven a una posición diferente en la estructura de extensión del conjunto de parámetros de vídeo. También se pretende que estos nuevos elementos de sintaxis se puedan mover a otra posición en la extensión del conjunto de parámetros de vídeo o en el conjunto de parámetros de vídeo o en alguna otra parte normativa del flujo de bits. Cabe señalar que en la figura 32 y la figura 33 los elementos de sintaxis max_dimension_id_minus1[i] se señalizan cuando vps_implicit_dimension_ids_flag tiene un valor de 1 o VERDADERO. Sin embargo, en otra realización, los elementos de sintaxis max_dimension_id_minus1[i] pueden señalizarse independientemente del valor del elemento de sintaxis vps_implicit_dimension_ids_flag.

La figura 33 y la figura 35 muestran una sintaxis variante adicional con una funcionalidad similar a la de la figura 32 y la figura 34 respectivamente. En la figura 33 y la figura 35 la marca vps_nuh_layer_id_present_flag y el elemento de sintaxis layer_id_in_nuh[i] se señalizan incluso cuando vps_implicit_dimension_ids_flag tiene un valor de 1. En la

figura 32 y la figura 34 la marca `vps_nuh_layer_id_present_flag` y el elemento de sintaxis `layer_id_in_nuh[i]` no se señalizan incluso cuando `vps_implicit_dimension_ids_flag` tiene un valor de 1.

La figura 36 y la figura 37 muestran una sintaxis variante adicional con una funcionalidad similar a la figura 32 y la figura 33 respectivamente. En la figura 36 y la figura 37 se señalizan elementos de sintaxis adicionales `dimension_index_order[i]`. `dimension_index_order[i]` especifica el valor de orden con respecto a dónde se señala la dimensión de escalabilidad particular `i` cuando se señalizan valores de `dimension_id` en un elemento de sintaxis contiguo. En este caso, cuando los valores de `dimension_id` se señalizan implícitamente, se infieren como `DimID[i][j]` como se describe en el código siguiente durante el proceso de decodificación.

```

10  int Id[]={0,0,0,...,0};
    for(i=0;i<=vps_max_layers_minus1;i++)
    {for(j=0;j<NumScalabilityTypes;j++)
      {
        DimId[i][dimension_index_order[j]]= Id[j];
      }
    Id[NumScalabilityTypes-1]= Id[NumScalabilityTypes-1]+1;
    for(k= NumScalabilityTypes-1 ;k>=0;k--)
    {
      if(Id[k]==(max_dimension_id_minus1[k]+1))
      {
        Id[k]=0;
        Id[k-1]= Id[k-1]+1;
      } } }

```

La figura 38 y la figura 39 muestran una sintaxis variante con funcionalidad similar a la de la sintaxis de la figura 36 y la figura 37 respectivamente. Todos los elementos de sintaxis de la figura 38 y la figura 39 tienen el mismo significado semántico que en la figura 36 y la figura 37, respectivamente. En la figura 38 y la figura 39 algunos de los elementos de sintaxis (`vps_implicit_dimension_ids_flag`, `max_dimension_id_minus1[i]`, `dimension_index_order[i]`) se mueven a una posición diferente en la estructura de extensión del conjunto de parámetros de video. También se pretende que estos nuevos elementos de sintaxis se puedan mover a otra posición en la extensión del conjunto de parámetros de video o en el conjunto de parámetros de video o en alguna otra parte normativa del flujo de bits.

En otra realización, el elemento de sintaxis `max_dimension_id_minus1[i]` se puede señalar usando `u(v)` en lugar de `u(3)`. En este caso, la longitud usada en la codificación `u(v)` será igual al valor del elemento de sintaxis correspondiente `dimension_id_len_minus1[i] + 1`.

En otra realización, el elemento de sintaxis `dimension_index_order[i]` se puede señalar usando `u(v)` en lugar de `u(3)`. En este caso, la longitud usada en la codificación `u(v)` será igual al número de bits necesarios para señalar el valor de `NumScalabilityTypes`.

En otra realización, uno o más de estos elementos de sintaxis `max_dimension_id_minus1[i]`, `dimension_index_order[i]` podrían indicarse con `ue(v)` o algún otro esquema de codificación.

En otra realización, se podrían señalar varios elementos de sintaxis nuevos en diferentes lugares dentro del VPS.

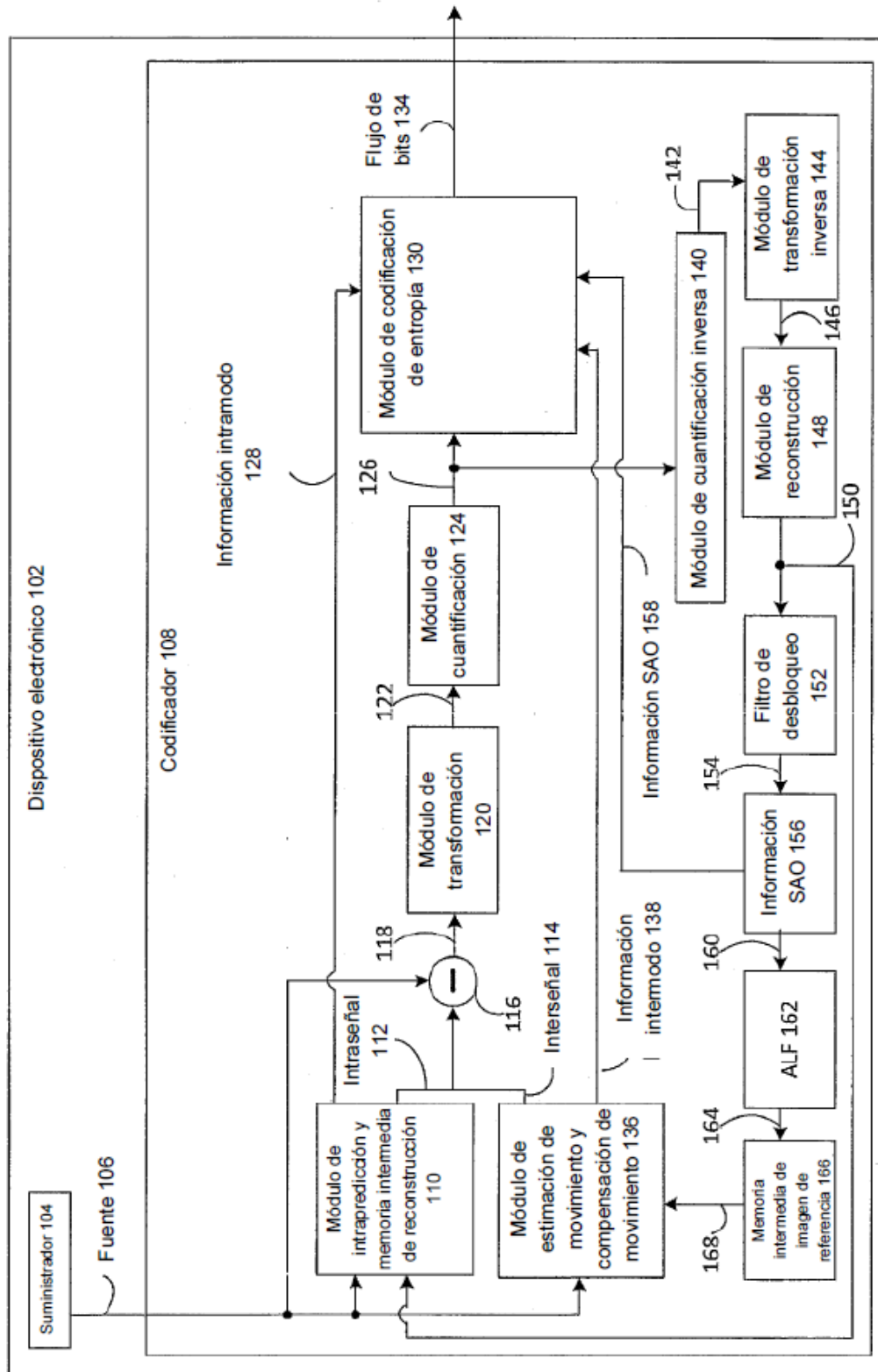
Los términos y expresiones que se han empleado en la especificación anterior se usan en la misma como términos de descripción y no de limitación, y no hay intención, en el uso de dichos términos y expresiones, de excluir equivalentes de las características mostradas y descritas o porciones de los mismos, reconociendo que el alcance de la invención está definido y limitado únicamente por las reivindicaciones que siguen.

REIVINDICACIONES

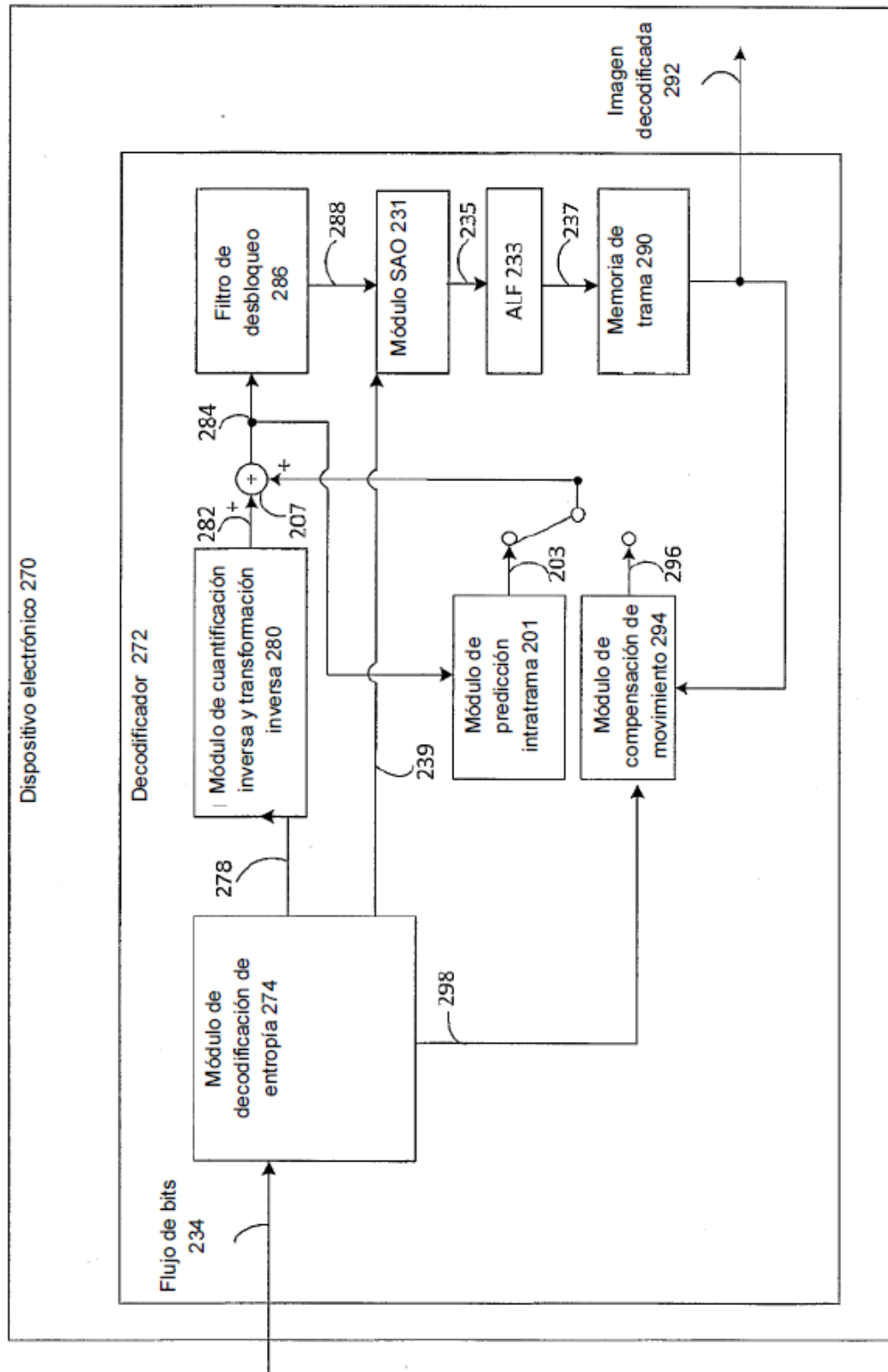
1.- Un método para decodificar un flujo de bits de video escalable codificado que comprende:

- 5 recibir un flujo de bits de video codificado que incluye una capa base, una o más capas de mejora y un conjunto de parámetros de video que comprende una información de escalabilidad sobre cada capa,
- en el que el conjunto de parámetros de video incluye:
- 10 una marca que indica si los identificadores de dimensión de escalabilidad se señalizan implícita o explícitamente; y
- un valor máximo de los identificadores de dimensión de escalabilidad para cada dimensión de escalabilidad de una pluralidad de dimensiones de escalabilidad para cada capa,
- 15 en el que la presencia o ausencia de la pluralidad de dimensiones de escalabilidad en la capa respectiva se indica en una máscara de escalabilidad señalizada en el flujo de bits,
- en el que cada identificador de dimensión de escalabilidad especifica la dimensión de escalabilidad de cada capa que incluye una o más capas de mejora y la capa base, siendo la dimensión de escalabilidad uno de múltiples tipos
- 20 que incluyen un tipo espacial y un tipo de calidad;
- determinar basándose en la marca si los identificadores de dimensión de escalabilidad están señalizados implícita o explícitamente;
- 25 decodificar el flujo de bits de video codificado usando la capa base y una o más capas de mejora, e
- inferir durante la decodificación y a partir de un elemento de sintaxis de un encabezado de unidad de capa de abstracción de red (NAL) en el flujo de bits de video codificado, una o más dimensiones de escalabilidad
- 30 identificadores de una unidad NAL, donde la unidad NAL corresponde al encabezado de la unidad NAL, en el que en respuesta a determinar que los identificadores de dimensión de escalabilidad están señalizados implícitamente, dicha inferencia se realiza implícitamente basándose en dicho valor máximo de los identificadores de dimensión de escalabilidad y dicho elemento de sintaxis.

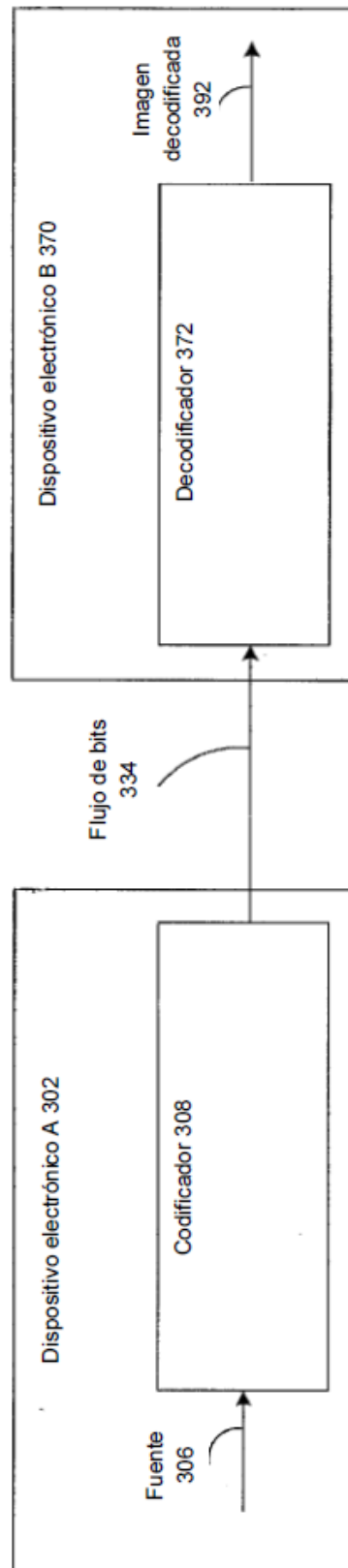
[Fig. 1]



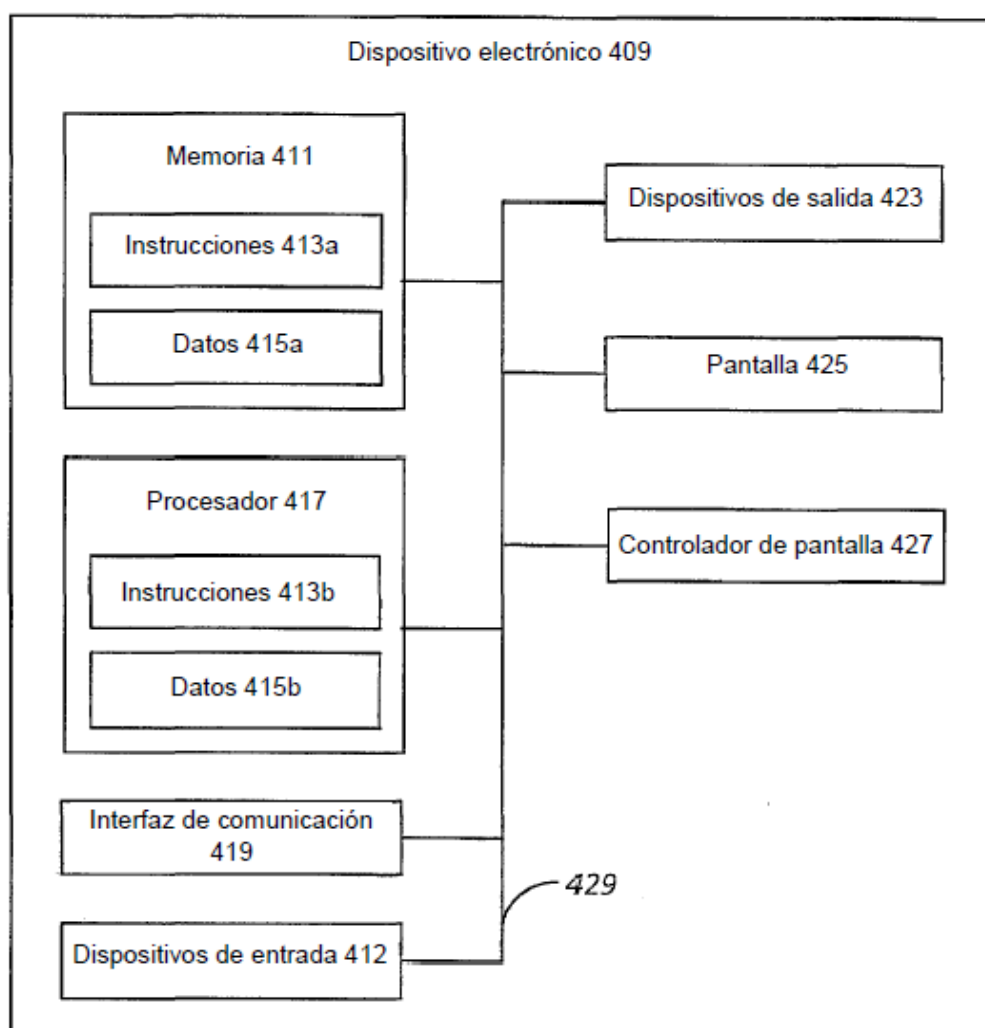
[Fig. 2]



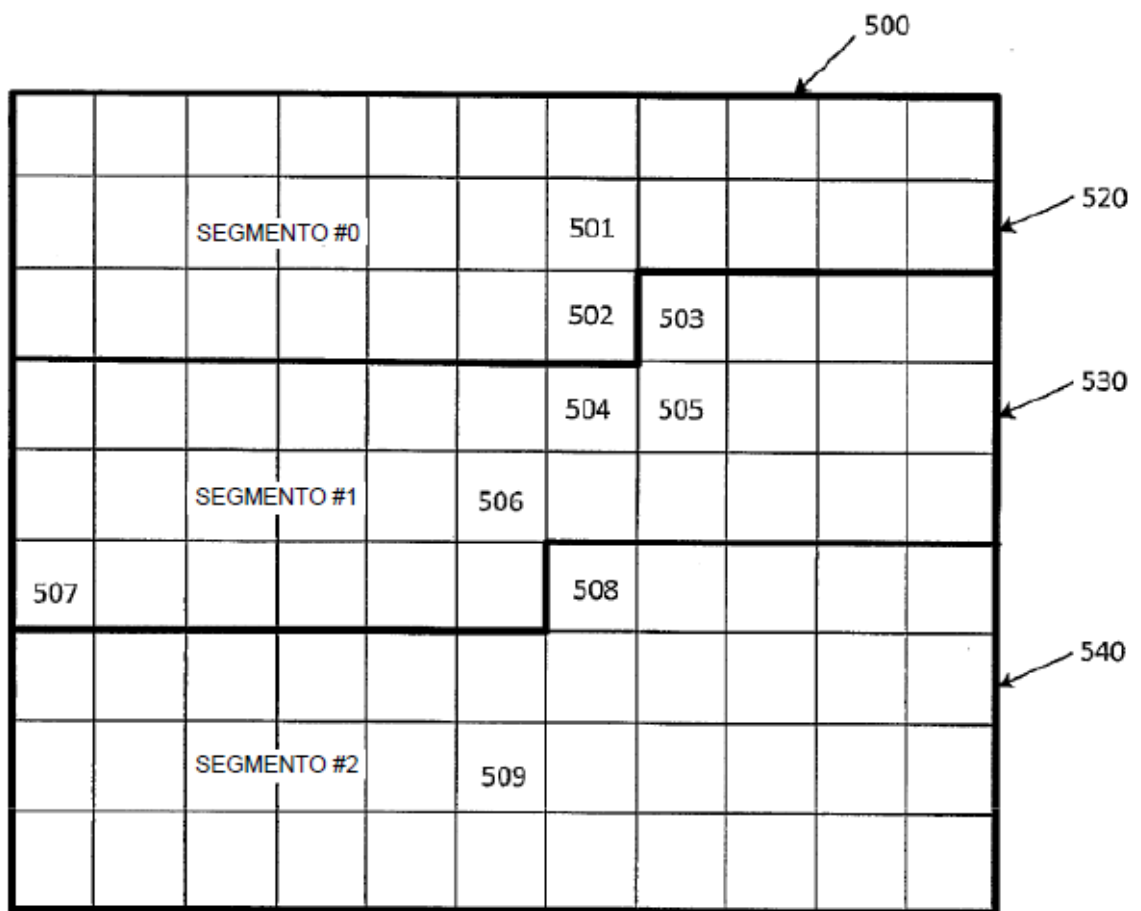
[Fig. 3]



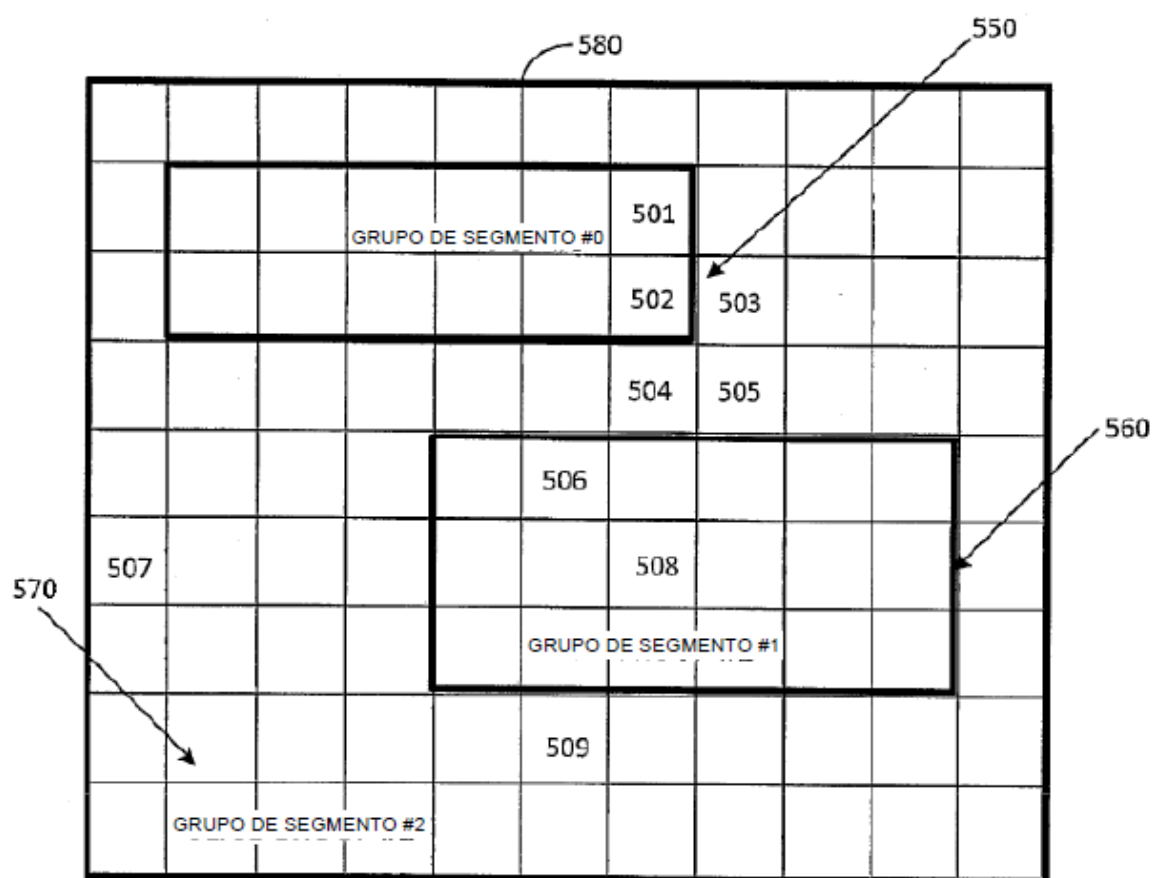
[Fig. 4]



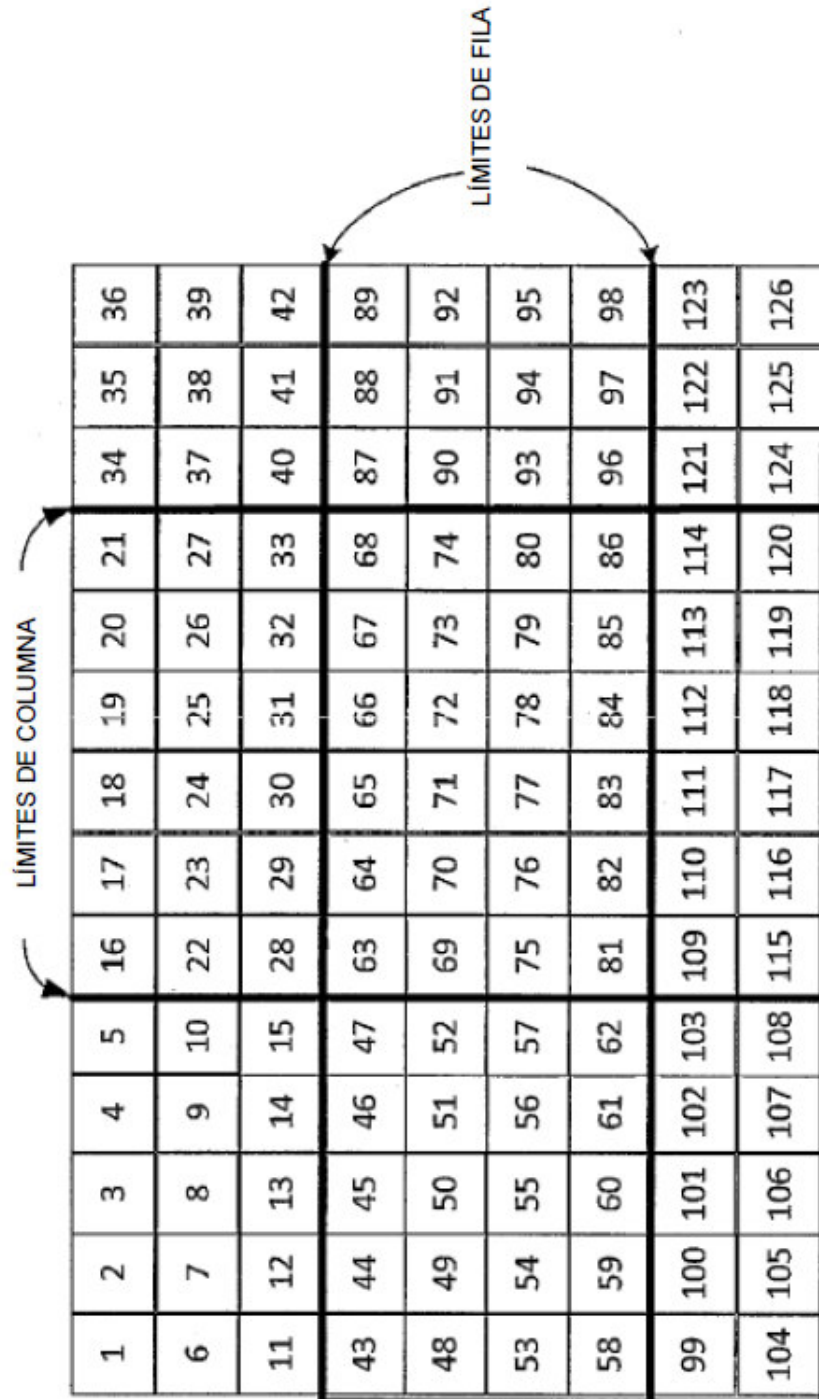
[Fig. 5]



[Fig. 6]



[Fig. 7]

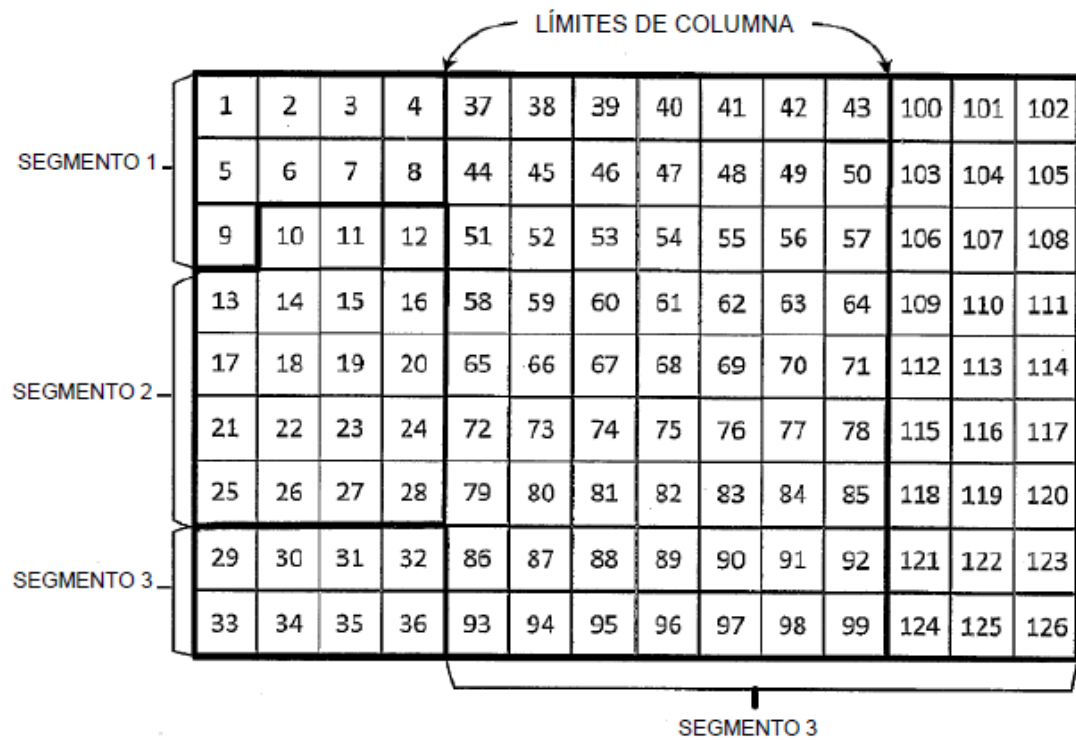


The figure shows a 10x10 grid of numbers. The rows are labeled with numbers 1 through 104, and the columns are labeled with numbers 1 through 104. The grid is divided into four quadrants by a vertical line between columns 5 and 6, and a horizontal line between rows 5 and 6. The numbers in the grid are as follows:

1	2	3	4	5	16	17	18	19	20	21	34	35	36
6	7	8	9	10	22	23	24	25	26	27	37	38	39
11	12	13	14	15	28	29	30	31	32	33	40	41	42
43	44	45	46	47	63	64	65	66	67	68	87	88	89
48	49	50	51	52	69	70	71	72	73	74	90	91	92
53	54	55	56	57	75	76	77	78	79	80	93	94	95
58	59	60	61	62	81	82	83	84	85	86	96	97	98
99	100	101	102	103	109	110	111	112	113	114	121	122	123
104	105	106	107	108	115	116	117	118	119	120	124	125	126

Arrows indicate the limits of the columns and rows. The label "LÍMITES DE COLUMNA" is positioned above the grid, and the label "LÍMITES DE FILA" is positioned to the right of the grid.

[Fig. 8]



[Fig. 9A]

nal_unit_header() {	Descriptor
forbidden_zero_bit	f(1)
nal_unit_type	u(6)
nuh_reserved_zero_6bits	u(6)
nuh_temporal_id_plus1	u(3)
}	

SINTAXIS DE ENCABEZADO DE UNIDAD NAL

[Fig. 9B]

nal_unit_header() {	Descriptor
forbidden_zero_bit	f(1)
nal_unit_type	u(6)
layer_id_plus1	u(6)
nuh_temporal_id_plus1	u(3)
}	

SINTAXIS DE ENCABEZADO DE UNIDAD NAL

[Fig. 9C]

nal_unit_header() {	Descriptor
forbidden_zero_bit	f(1)
nal_unit_type	u(6)
layer_id	u(6)
nuh_temporal_id_plus1	u(3)
}	

SINTAXIS DE ENCABEZADO DE UNIDAD NAL

[Fig. 10]

nal_unit(NumBytesInNALunit) {	Descriptor
nal_unit_header() .	
NumBytesInRBSP = 0	
for(i = 2; i < NumBytesInNALunit; i++) {	
if(i + 2 < NumBytesInNALunit && next_bits(24) == 0x000003) {	
rbsp_byte[NumBytesInRBSP++]	b(8)
rbsp_byte[NumBytesInRBSP++]	b(8)
i += 2	
emulation_prevention_three_byte /* equal to 0x03 */	f(8)
} else	
rbsp_byte[NumBytesInRBSP++]	b(8)
}	
}	

SINTAXIS DE ENCABEZADO DE UNIDAD NAL

[Fig. 11]

vps_extension() {	Descriptor
while(!byte_aligned())	
vps_extension_byte_alignment_reserved_zero_bit	u(1)
// scalability type and layer_id partitioning method	
scalability_type	u(4)
for(i = 0; i < MaxDim(scalability_type); i++)	
layer_id_dim_len[i]	u(3)
// layer specific information	
for(i = 0; i <= max_num_layers_minus1; i++) {	
vps_layer_id[i]	u(6)
// layer dependency	
num_direct_ref_layers[i]	u(6)
for(j = 0; j < num_direct_ref_layers[i]; j++)	
ref_layer_id[i][j]	u(6)
}	
}	

Sintaxis de detección de conjunto de parámetros de video existente

[Fig. 12]

scalability_type	MaxDim(scalability_type)	Dimensiones de escalabilidad
0	1	ninguna (HEVC base)
1	2	espacial y calidad
2	3	espacial, calidad, sin especificar
3	4	espacial, calidad, sin especificar, sin especificar
4	2	multivista y profundidad
5	3	multivista, profundidad, sin especificar
6	4	multivista, profundidad, sin especificar, sin especificar
7	4	multivista, espacial, calidad y profundidad
8	5	multivista, espacial, calidad, profundidad, sin especificar
9	6	multivista, espacial, calidad, profundidad, sin especificar, sin especificar
10...15	reservada	reservada

Tipos de escalabilidad existentes

[Fig. 13]

vps_extension() {	Descriptor
while(!byte_aligned())	
vps_extension_byte_alignment_reserved_zero_bit	u(1)
// Scalability map and layer_id_plus1 mapping method	
for(i=0; i<= max_num_layers_minus1_bits; i++) {	
scalability_map[i]	u(3)
}	
// layer specific information	
for(i = 0; i <= max_num_layers_minus1; i++) {	
vps_layer_id[i]	u(6)
// layer dependency	
num_direct_ref_layers[i]	u(6)
for(j = 0; j < num_direct_ref_layers[i]; j++)	
ref_layer_id[i][j]	u(6)
}	
}	

Sintaxis de extensión de conjunto de parámetros de video

[Fig. 14]

Scalability_map[i]	Dimensión de escalabilidad
0	ninguna (HEVC base)
1	espacial
2	calidad
3	profundidad
4	multivista
5	sin especificar
6, 7	reservada

Sintaxis de mapa de escalabilidad

[Fig. 15]

vps_extension() {	Descriptor
while(!byte_aligned())	
vps_extension_byte_alignment_reserved_zero_bit	u(1)
// scalability map and layer_id_plus1 mapping method	
num_scalability_dimensions_minus1	u(3)
for(i=0; i<= num_scalability_dimensions_minus1; i++) {	
scalability_map[i]	u(3)
num_bits_for_scalability_map[i]	u(3)
}	
// layer specific information	
for(i = 0; i <= max_num_layers_minus1; i++) {	
vps_layer_id[i]	u(6)
// layer dependency	
num_direct_ref_layers[i]	u(6)
for(j = 0; j < num_direct_ref_layers[i]; j++)	
ref_layer_id[i][j]	u(6)
}	
}	

Sintaxis de extensión de conjunto de parámetros de vídeo

[Fig. 16]

vps_extension() {	Descriptor
while(!byte_aligned())	
vps_extension_byte_alignment_reserved_zero_bit	u(1)
// layer specific information	
for(i = 1; i <= vps_max_layers_minus1; i++) {	
// mapping of layer ID to scalability dimension IDs	
num_dimensions_minus1[i]	u(4)
for(j = 0; j <= num_dimensions_minus1; j++) {	
dimension_type[i][j]	u(4)
dimension_id[i][j]	u(8)
}	
// layer dependency	
num_direct_ref_layers[i]	u(6)
for(j = 0; j < num_direct_ref_layers[i]; j++)	
ref_layer_id[i][j]	u(6)
}	
}	

Sintaxis de extensión de conjunto de parámetros de vídeo existente

[Fig. 17]

dimension_type[i][j]	dimension_id[i][j]
0	orden de vista idx
1	marca de profundidad
2	ID de dependencia
3	ID de calidad
4..15	reservado

Tipo de dimensión existente, sintaxis de ID de dimensión

[Fig. 18]

vps_extension() {	Descriptor
while(!byte_aligned())	
vps_extension_byte_alignment_reserved_zero_bit	u(1)
// layer specific information	
for(i = 1; i <= max_num_layers_minus1; i++) {	
// mask signalling scalability types that are present for this layer ID	
scalability_mask	u(8)
for(j = 0; j <= num_scalability_types[i]; j++) {	
scalability_id[j]	u(8)
}	
// layer dependency	
num_direct_ref_layers[i]	u(6)
for(j = 0; j < num_direct_ref_layers[i]; j++)	
ref_layer_id[i][j]	u(6)
}	
}	

Sintaxis de extensión de conjunto de parámetros de vídeo

[Fig. 19]

scalability_mask_bit[i]	Dimensión de escalabilidad
0	ninguna (HEVC base)
1	espacial
2	calidad
3	profundidad
4	multivista
5	sin especificar
6	reservada
7	reservada

Sintaxis de mapa de escalabilidad

[Fig. 20]

video_parameter_set_rbsp() {	Descriptor
video_parameter_set_id	u(4)
vps_temporal_id_nesting_flag	u(1)
reserved_zero_2bits	u(2)
max_num_layers_minus1 // reserved_zero_6bits	u(6)
vps_max_sub_layers_minus1	u(3)
profile_level(1, vps_max_sub_layers_minus1)	
next_essential_info_byte_offset // reserved_zero_12bits	u(12)
for(i = 0; i <= vps_max_sub_layers_minus1; i++) {	
vps_max_dec_pic_buffering[i]	ue(v)
vps_max_num_reorder_pics[i]	ue(v)
vps_max_latency_increase[i]	ue(v)
}	
num_hrd_parameters	ue(v)
for(i = 0; i < num_hrd_parameters; i++) {	
if(i > 0)	
op_point(i)	
hrd_parameters(i == 0, vps_max_sub_layers_minus1)	
}	
bit_equal_to_one	u(1)
scalability_mask	u(8)
for(i = 0; i <= max_num_layers_minus1; i++) {	
vps_extension_data()	
}	
vps_extension_flag	u(1)
if(vps_extension_flag)	
while(more_rbsp_data())	
vps_extension_data_flag	u(1)
rbsp_trailing_bits()	
}	
}	

Sintaxis de extensión de conjunto de parámetros de vídeo

[Fig. 21]

vps_extension_data() {	Descriptor
vps_extension_type	u(2)
layer_id	u(6)
layer_max_sub_layers_minus1	u(3)
new_profile_level_flag	u(1)
if (new_profile_level_flag == 0) {	
profile_reference_layer_id	u(6)
}	
else {	
profile_level (1, layer_max_sub_layers_minus1)	
}	
for(j = 0; j <= num_scalability_types; j++) {	
scalable_id [j]	u(8)
}	
num_reference_layers	u(6)
for(n = 0; n < num_reference_layers; n ++)	
direct_coding_dependency_layer_id_plus1 [n]	u(6)
}	

Sintaxis de extensión de conjunto de parámetros de vídeo

[Fig. 22]

vps_extension() {	Descriptor
while(!byte_aligned())	
vps_extension_byte_aligment_reserved_one_bit	u(1)
avc_base_codec_flag	u(1)
scalability_mask	u(16)
for(i = 0; i < NumScalabilityTypes; i++) {	
dimension_id_len_minus1[i]	u(3)
}	
vps_nuh_layer_id_present_flag	u(1)
// layer specific information	
for(i = 1; i <= vps_max_layers_minus1; i++) {	
// mapping of layer ID to scalability dimension IDs	
if(vps_nuh_layer_id_present_flag)	
layer_id_in_nuh[i]	u(6)
for(j = 0; j <= num_dimensions_minus1; j++)	
dimension_id[i][j]	u(v)
source_scan_type_info_idc[i]	u(2)
source_2d_3d_info_idc[i]	u(2)
}	
for(i = 1; i <= vps_max_layers_minus1; i++)	
profile_tier_level(i, vps_max_sub_layers_minus1)	
for(i = 1; i <= vps_max_layers_minus1; i++) {	
// layer dependency information signaling	
num_direct_ref_layers[i]	u(6)
for(j = 0; j < num_direct_ref_layers[i]; j++)	
ref_layer_id[i][j]	u(6)
}	
}	

Sintaxis de extensión de conjunto de parámetros de vídeo

[Fig. 23]

source_scan_type_info_idc[i]	Características de capa de vídeo
0	Todas las imágenes en capa i tienen un tipo de escaneo entrelazado
1	Todas las imágenes en capa i tienen un tipo de escaneo progresivo
2	Las imágenes en capa i tienen un tipo de escaneo desconocido
3	Las imágenes en capa i tienen una mezcla de tipo de escaneo entrelazado y progresivo

Sintaxis de indicador de información de tipo de escaneo de fuente

[Fig. 24]

source_2d_3d_info_idc[i]	Características de capa de vídeo
0	Todas las imágenes en la capa i son tramas 2D
1	Todas las imágenes en la capa i están empaquetadas en formato 3D compatible con trama
2	Las imágenes en la capa i tienen una disposición de empaquetado de trama desconocida
3	Las imágenes en la capa i tienen una mezcla de tramas 3D y 2D compatibles con trama

Sintaxis de indicador de información 2D, 3D de fuente

[Fig. 25]

vps_extension() {	Descriptor
while(!byte_aligned())	
vps_extension_byte_alignment_reserved_one_bit	u(1)
avc_base_codec_flag	u(1)
scalability_mask	u(16)
for(i = 0; i < NumScalabilityTypes; i++) {	
dimension_id_len_minus1[i]	u(3)
}	
source_scan_type_info_idc	u(2)
source_2d_3d_info_idc	u(2)
vps_nuh_layer_id_present_flag	u(1)
// layer specific information	
for(i = 1; i <= vps_max_layers_minus1; i++) {	
// mapping of layer ID to scalability dimension IDs	
if(vps_nuh_layer_id_present_flag)	
layer_id_in_nuh[i]	u(6)
for(j = 0; j <= num_dimensions_minus1; j++)	
dimension_id[i][j]	u(v)
}	
for(i = 1; i <= vps_max_layers_minus1; i++)	
profile_tier_level(1, vps_max_sub_layers_minus1)	
for(i = 1; i <= vps_max_layers_minus1; i++) {	
// layer dependency information signaling	
num_direct_ref_layers[i]	u(6)
for(j = 0; j < num_direct_ref_layers[i]; j++)	
ref_layer_id[i][j]	u(6)
}	
}	

Sintaxis de extensión de conjunto de parámetros de vídeo

[Fig. 26]

video_parameter_set_rbsp() {	Descriptor:
video_parameter_set_id	u(4)
vps_temporal_id_nesting_flag	u(1)
vps_reserved_zero_2bits	u(2)
vps_max_num_layers_minus1 //reserved_zero_6bits in the base spec	u(6)
vps_max_sub_layers_minus1	u(3)
next_essential_info_byte_offset //vps_reserved_zero_16bits in the base spec	u(16)
profile_tier_level(l, vps_max_sub_layers_minus1)	
for(i = 0; i <= vps_max_sub_layers_minus1; i++) {	
vps_max_dec_pic_buffering[i]	ue(v)
vps_max_num_reorder_pics[i]	ue(v)
vps_max_latency_increase[i]	ue(v)
}	
source_scan_type_info_idc	u(2)
source_2d_3d_info_idc	u(2)
vps_num_hrd_parameters	ue(v)
vps_max_nuh_reserved_zero_layer_id	u(6)
for(i = 0; i < vps_num_hrd_parameters; i++) {	
if(i > 0)	
operation_point_layer_ids(i)	
hrd_parameters(i == 0, vps_max_sub_layers_minus1)	
}	
bit_equal_to_one	u(1)
vps_extension()	
vps_extension_flag	u(1)
if(vps_extension_flag)	
while(more_rbsp_data())	
vps_extension_data_flag	u(1)
rbsp_trailing_bits()	
}	

Sintaxis de conjunto de parámetros de vídeo

[Fig. 27]

source_scan_type_info_idc[i]	Características de capa de vídeo
0	Todas las imágenes en la capa i tienen un tipo de escaneo entrelazado
1	Todas las imágenes en la capa i tienen un tipo de escaneo progresivo
2	Las imágenes en la capa i tienen un tipo de escaneo desconocido
3	Las imágenes en la capa i tienen una mezcla de tipo de escaneo entrelazado y progresivo
4-7	Reservada

Sintaxis de indicador de información de tipo de escaneo de fuente

[Fig. 28]

source 2d 3d info idc[i]	Características de capa de video
0	Todas las imágenes en la capa i son tramas 2D
1	Todas las imágenes en la capa i están empaquetadas en formato 3D compatible con trama
2	Las imágenes en la capa i tienen una disposición de empaquetado desconocido
3	Las imágenes en la capa i tienen una mezcla tramas 3D y 2D compatibles con trama
4-7	Reservada

Sintaxis de indicador de información 2D, 3D de fuente

[Fig. 29]

scalability_mask_bit[k]	Dimensión de escalabilidad
0	espacial
1	calidad
2	profundidad
3	multivista
4-15	reservada

Sintaxis de mapa de escalabilidad

[Fig. 30]

vps_extension() {	Descriptor
while(!byte_aligned())	
vps_extension_byte_alignment_reserved_one_bit	u(1)
avc_base_codec_flag	u(1)
scalability_mask	u(1-6)
for(i = 0; i < NumScalabilityTypes; i++) {	
dimension_id_len_minus1[i]	v(3)
}	
vps_nuh_layer_id_present_flag	u(1)
// layer specific information	
for(i = 1; i <= vps_max_layers_minus1; i++) {	
// mapping of layer ID to scalability dimension IDs	
if(vps_nuh_layer_id_present_flag)	
layer_id_in_nuh[i]	u(6)
for(j = 0; j < NumScalabilityTypes; j++)	
dimension_id[i][j]	u(v8)
}	
for(i = 1; i <= vps_max_layers_minus1; i++)	
profile_for_level(1, vps_max_sub_layers_minus1)	
for(i = 1; i <= vps_max_layers_minus1; i++) {	
// layer dependency	
num_direct_ref_layers[i]	u(6)
for(j = 0; j < num_direct_ref_layers[i]; j++)	
ref_layer_id[i][j]	u(6)
}	
}	

Sintaxis de extensión de conjunto de parámetros de vídeo

[Fig. 31]

scalability_mask's k'th bit	Dimensión de escalabilidad
0	espacial
1	calidad
2	profundidad
3	multivista
4-15	reservada

Sintaxis scalability_mask

[Fig. 32]

vps_extension() {	Descriptor
while(!byte_aligned())	
vps_extension_byte_alignment_reserved_one_bit	u(1)
avc_base_codec_flag	u(1)
scalability_mask	u(16)
vps_implicit_dimension_ids_flag	u(1)
for(i = 0; i < NumScalabilityTypes; i++) {	
dimension_id_len_minus1[i]	u(3)
if(vps_implicit_dimension_ids_flag)	
max_dimension_id_minus1[i]	u(3)
}	
if(!vps_implicit_dimension_ids_flag) {	
vps_nuh_layer_id_present_flag	u(1)
// layer specific information	
for(i = 1; i <= vps_max_layers_minus1; i++) {	
// mapping of layer ID to scalability dimension IDs	
if(vps_nuh_layer_id_present_flag)	
layer_id_in_nuh[i]	u(6)
for(j = 0; j <= num_dimensions_minus1; j++)	
dimension_id[i][j]	u(v)
}	
}	
for(i = 1; i <= vps_max_layers_minus1; i++)	
profile_tier_level(1, vps_max_sub_layers_minus1)	
for(i = 1; i <= vps_max_layers_minus1; i++) {	
// layer dependency information signaling	
num_direct_ref_layers[i]	u(6)
for(j = 0; j < num_direct_ref_layers[i]; j++)	
ref_layer_id[i][j]	u(6)
}	
}	

Sintaxis de extensión de conjunto de parámetros de vídeo

[Fig. 33]

vps_extension() {	Descriptor
while(!byte_aligned())	
vps_extension_byte_alignment_reserved_one_bit	u(1)
avc_base_codec_flag	u(1)
scalability_mask	u(16)
vps_implicit_dimension_ids_flag	u(1)
for(i = 0; i < NumScalabilityTypes; i++) {	
dimension_id_len_minus1[i]	u(3)
if(vps_implicit_dimension_ids_flag)	
max_dimension_id_minus1[i]	u(3)
}	
vps_nuh_layer_id_present_flag	u(1)
// layer specific information	
for(i = 1; i <= vps_max_layers_minus1; i++) {	
// mapping of layer ID to scalability dimension IDs	
if(vps_nuh_layer_id_present_flag)	
layer_id_in_nuh[i]	u(6)
if(!vps_implicit_dimension_ids_flag) {	
for(j = 0; j <= num_dimensions_minus1; j++)	
dimension_id[i][j]	u(v)
}	
}	
for(i = 1; i <= vps_max_layers_minus1 ; i++)	
profile_tier_level(1, vps_max_sub_layers_minus1)	
for(i = 1; i <= vps_max_layers_minus1; i++) {	
// layer dependency information signaling	
num_direct_ref_layers[i]	u(6)
for(j = 0; j < num_direct_ref_layers[i]; j++)	
ref_layer_id[i][j]	u(6)
}	
}	

Sintaxis de extensión de conjunto de parámetros de vídeo

[Fig. 34]

vps_extension() {	Descriptor
while(!byte_aligned())	
vps_extension_byte_alignment_reserved_one_bit	u(1)
avc_base_codec_flag	u(1)
scalability_mask	u(16)
for(i = 0; i < NumScalabilityTypes; i++) {	
dimension_id_len_minus1[i]	u(3)
}	
vps_implicit_dimension_ids_flag	u(1)
if(vps_implicit_dimension_ids_flag) {	
for(i = 0; i < NumScalabilityTypes; i++)	
max_dimension_id_minus1[i]	u(3)
}	
else{	
vps_nuh_layer_id_present_flag	u(1)
// layer specific information	
for(i = 1; i <= vps_max_layers_minus1; i++) {	
// mapping of layer ID to scalability dimension IDs	
if(vps_nuh_layer_id_present_flag)	
layer_id_in_nuh[i]	u(6)
for(j = 0; j <= num_dimensions_minus1; j++)	
dimension_id[i][j]	u(v)
}	
}	
for(i = 1; i <= vps_max_layers_minus1; i++)	
profile_tier_level(1, vps_max_sub_layers_minus1)	
for(i = 1; i <= vps_max_layers_minus1; i++) {	
// layer dependency information signaling	
num_direct_ref_layers[i]	u(6)
for(j = 0; j < num_direct_ref_layers[i]; j++)	
ref_layer_id[i][j]	u(6)
}	
}	

Sintaxis de extensión de conjunto de parámetros de vídeo

[Fig. 35]

vps_extension() {	Descriptor
while(!byte_aligned())	
vps_extension_byte_alignment_reserved_one_bit	u(1)
avc_base_codec_flag	u(1)
scalability_mask	u(16)
for(i = 0; i < NumScalabilityTypes; i++) {	
dimension_id_len_minus1[i]	u(3)
}	
vps_implicit_dimension_ids_flag	u(1)
if(vps_implicit_dimension_ids_flag) {	
for(i = 0; i < NumScalabilityTypes; i++)	
max_dimension_id_minus1[i]	u(3)
}	
vps_nuh_layer_id_present_flag	u(1)
// layer specific information	
for(i = 1; i <= vps_max_layers_minus1; i++) {	
// mapping of layer ID to scalability dimension IDs	
if(vps_nuh_layer_id_present_flag)	
layer_id_in_nuh[i]	u(6)
if(!vps_implicit_dimension_ids_flag) {	
for(j = 0; j <= num_dimensions_minus1; j++)	
dimension_id[i][j]	u(v)
}	
}	
for(i = 1; i <= vps_max_layers_minus1; i++)	
profile_tier_level(1, vps_max_sub_layers_minus1)	
for(i = 1; i <= vps_max_layers_minus1; i++) {	
// layer dependency information signaling	
num_direct_ref_layers[i]	u(6)
for(j = 0; j < num_direct_ref_layers[i]; j++)	
ref_layer_id[i][j]	u(6)
}	
}	

Sintaxis de extensión de conjunto de parámetros de vídeo

[Fig. 36]

vps_extension() {	Descriptor
while(!byte_aligned())	
vps_extension_byte_alignment_reserved_one_bit	u(1)
avc_base_codec_flag	u(1)
scalability_mask	u(16)
vps_implicit_dimension_ids_flag	u(1)
for(i = 0; i < NumScalabilityTypes; i++) {	
dimension_id_len_minus1[i]	u(3)
if(vps_implicit_dimension_ids_flag) {	
max_dimension_id_minus1[i]	u(3)
dimension_index_order[i]	u(4)
}	
}	
if(!vps_implicit_dimension_ids_flag) {	
vps_nuh_layer_id_present_flag	u(1)
// layer specific information	
for(i = 1; i <= vps_max_layers_minus1; i++) {	
// mapping of layer ID to scalability dimension IDs	
if(vps_nuh_layer_id_present_flag)	
layer_id_in_nuh[i]	u(6)
for(j = 0; j <= num_dimensions_minus1; j++)	
dimension_id[i][j]	u(v)
}	
}	
for(i = 1; i <= vps_max_layers_minus1; i++)	
profile_tier_level(1, vps_max_sub_layers_minus1)	
for(i = 1; i <= vps_max_layers_minus1; i++) {	
// layer dependency information signaling	
num_direct_ref_layers[i]	u(6)
for(j = 0; j < num_direct_ref_layers[i]; j++)	
ref_layer_id[i][j]	u(6)
}	
}	

Sintaxis de extensión de conjunto de parámetros de vídeo

[Fig. 37]

vps_extension() {	Descriptor
while(!byte_aligned())	
vps_extension_byte_alignment_reserved_one_bit	u(1)
avc_base_codec_flag	u(1)
scalability_mask	u(16)
vps_implicit_dimension_ids_flag	u(1)
for(i = 0; i < NumScalabilityTypes; i++) {	
dimension_id_len_minus1[i]	u(3)
if(vps_implicit_dimension_ids_flag) {	
max_dimension_id_minus1[i]	u(3)
dimension_index_order[i]	u(4)
}	
}	
vps_nuh_layer_id_present_flag	u(1)
// layer specific information	
for(i = 1; i <= vps_max_layers_minus1; i++) {	
// mapping of layer ID to scalability dimension IDs	
if(vps_nuh_layer_id_present_flag)	
layer_id_in_nuh[i]	u(6)
if(!vps_implicit_dimension_ids_flag) {	
for(j = 0; j <= num_dimensions_minus1; j++)	
dimension_id[i][j]	u(v)
}	
}	
for(i = 1; i <= vps_max_layers_minus1 ; i++)	
profile_tier_level(1, vps_max_sub_layers_minus1)	
for(i = 1; i <= vps_max_layers_minus1; i++) {	
// layer dependency information signaling	
num_direct_ref_layers[i]	u(6)
for(j = 0; j < num_direct_ref_layers[i]; j++)	
ref_layer_id[i][j]	u(6)
}	
}	

Sintaxis de extensión de conjunto de parámetros de vídeo

[Fig. 38]

vps_extension() {	Descriptor
while(lbyte_aligned())	
vps_extension_byte_aligment_reserved_one_bit	u(1)
avc_base_codec_flag	u(1)
scalability_mask	u(16)
for(i = 0; i < NumScalabilityTypes; i++) {	
dimension_id_len_minus1[i]	u(3)
}	
vps_implicit_dimension_ids_flag	u(1)
if(vps_implicit_dimension_ids_flag) {	
for(i = 0; i < NumScalabilityTypes; i++)	
max_dimension_id_minus1[i]	u(3)
dimension_index_order[i]	u(4)
}	
else{	
vps_nuh_layer_id_present_flag	u(1)
// layer specific information	
for(i = 1; i <= vps_max_layers_minus1; i++) {	
// mapping of layer ID to scalability dimension IDs	
if(vps_nuh_layer_id_present_flag)	
layer_id_in_nuh[i]	u(6)
for(j = 0; j <= num_dimensions_minus1; j++)	
dimension_id[i][j]	u(v)
}	
}	
for(i = 1; i <= vps_max_layers_minus1; i++)	
profile_tier_level(1, vps_max_sub_layers_minus1)	
for(i = 1; i <= vps_max_layers_minus1; i++) {	
// layer dependency information signaling	
num_direct_ref_layers[i]	u(6)
for(j = 0; j < num_direct_ref_layers[i]; j++)	
ref_layer_id[i][j]	u(6)
}	
}	

Sintaxis de extensión de conjunto de parámetros de vídeo

[Fig. 39]

vps_extension() {	Descriptor
while(!byte_aligned())	
vps_extension_byte_alignment_reserved_one_bit	u(1)
avc_base_codec_flag	u(1)
scalability_mask	u(16)
for(i = 0; i < NumScalabilityTypes; i++) {	
dimension_id_len_minus1[i]	u(3)
}	
vps_implicit_dimension_ids_flag	u(1)
if(vps_implicit_dimension_ids_flag) {	
for(i = 0; i < NumScalabilityTypes; i++)	
max_dimension_id_minus1[i]	u(3)
dimension_index_order[i]	u(4)
}	
vps_nuh_layer_id_present_flag	u(1)
// layer specific information	
for(i = 1; i <= vps_max_layers_minus1; i++) {	
// mapping of layer ID to scalability dimension IDs	
if(vps_nuh_layer_id_present_flag)	
layer_id_in_nuh[i]	u(6)
if(!vps_implicit_dimension_ids_flag) {	
for(j = 0; j <= num_dimensions_minus1; j++)	
dimension_id[i][j]	u(v)
}	
}	
for(i = 1; i <= vps_max_layers_minus1; i++)	
profile_tier_level(1, vps_max_sub_layers_minus1)	
for(i = 1; i <= vps_max_layers_minus1; i++) {	
// layer dependency information signaling	
num_direct_ref_layers[i]	u(6)
for(j = 0; j < num_direct_ref_layers[i]; j++)	
ref_layer_id[i][j]	u(6)
}	
}	

Sintaxis de extensión de conjunto de parámetros de vídeo