(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 715 782**

(51) Int. CI.:

H04N 19/117 (2014.01)
H04N 19/82 (2014.01)
H04N 19/70 (2014.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **20.04.2012 E 17169267 (6)**

(97) Fecha y número de publicación de la concesión europea: **02.01.2019 EP 3220641**

(54) Título: **Procedimiento y aparato para un filtrado en bucle mejorado**

(30) Prioridad:

21.04.2011 US 201161477689 P
14.10.2011 US 201161547281 P
07.02.2012 US 201261595914 P
07.02.2012 US 201261595900 P
13.02.2012 US 201261597995 P
17.02.2012 US 201261600028 P

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
06.06.2019

(73) Titular/es:

HFI INNOVATION INC. (100.0%)
3F.-7, No.5, Taiyuan 1st Street, Zhubei City
Hsinchu County 302, TW

(72) Inventor/es:

FU, CHIH-MING;
CHEN, CHING-YEH;
TSAI, CHIA-YANG;
HUANG, YU-WEN y
LEI, SHAW-MIN

(74) Agente/Representante:

CONTRERAS PÉREZ, Yahel

DESCRIPCIÓN

Procedimiento y aparato para un filtrado en bucle mejorado.

La presente invención reivindica prioridad de la Solicitud de Patente Provisional de EE.UU., número de serie 61/477.689, presentada el 21 de abril de 2011, titulada "Improved Sample Adaptive Offset", la Solicitud de Patente Provisional de EE.UU., número de serie 61/547.281, presentada el 14 de octubre de 2011, titulada "Low Latency Loop Filtering", la Solicitud de Patente Provisional de EE.UU., número de serie 61/595.900, presentada el 17 de febrero de 2012, titulada "Improved Sample Adaptive Offset", la Solicitud de patente provisional de EE.UU., número de serie 61/595.914, presentada el 7 de febrero de 2012, titulada "Improved LCU-based Encoding Algorithm of ALF", la Solicitud de patente provisional de EE.UU., número de serie 61/597.995, presentada el 13 de febrero de 2012, titulada "Improved ALF and SAO", y la Solicitud de patente provisional de EE.UU., número de serie 61/600.028, presentada el 17 de febrero de 2012, titulada "LCU-based Syntax for SAO and ALF".

CAMPO TÉCNICO

La presente invención se refiere a un sistema de codificación de vídeo. En particular, la presente invención se refiere a un procedimiento y aparato para un procesamiento en bucle tal como SAO y ALF mejorado. El uso de procedimientos SAO después del desbloqueo en la codificación de vídeo se divulga en "CE13: Sample Adaptive Offset with LCU-Independent Decoding", JCTVC-E049, 10 de marzo de 2011 (10-03-2011).

ANTECEDENTES

La estimación de movimiento es una técnica eficaz de codificación entre fotogramas para explotar la redundancia temporal en las secuencias de vídeo. La codificación entre fotogramas de movimiento compensado se ha usado ampliamente en diversos estándares de codificación de video internacionales. La estimación de movimiento adoptada en diversos estándares de codificación es a menudo una técnica basada en bloques, en el que la información de movimiento, tal como el modo de codificación y el vector de movimiento, se determina para cada macro-bloque o configuración de bloque similar. Además, la intra-codificación también se aplica de forma adaptativa cuando la imagen es procesada sin referencia a ninguna otra imagen. Los residuos inter-pronosticados o intra-pronosticados son procesados normalmente por transformación, cuantificación y codificación de entropía para generar un flujo de bits de vídeo comprimido. Durante el proceso de codificación, se introducen artefactos de codificación, especialmente en el proceso de cuantificación. Con el fin de atenuar los artefactos de codificación, se ha aplicado un procesamiento adicional al vídeo reconstruido para mejorar la calidad de imagen en sistemas de codificación más recientes. El procesamiento adicional a menudo se configura en una operación en bucle de modo que el codificador y el decodificador puedan obtener las mismas imágenes de referencia para lograr un mejor rendimiento del sistema.

La Figura 1A ilustra un sistema de ejemplo de inter/intra codificación de vídeo adaptativa que incorpora el procesamiento en bucle. Para la inter predicción, se utiliza Estimación de Movimiento (ME)/Compensación de movimiento (MC) 112 para proporcionar datos de predicción en base a datos de vídeo de otra imagen o imágenes. Un conmutador 114 selecciona datos de Intra Predicción 110 o de inter-predicción y los datos de predicción seleccionados se suministran al Agregador 116 para formar errores de predicción, también denominados residuos. El error de predicción es entonces procesado por Transformación (T) 118 seguido por Cuantificación (Q) 120. Los residuos transformados y cuantificados son codificados por un Codificador de Entropía 122 para formar un flujo de vídeo correspondiente a los datos de vídeo comprimidos. El flujo de bits asociado con coeficientes de transformación es empaquetado con información colateral tal como de movimiento, modo y otra información asociada con el área de la imagen. La información colateral también puede ser sometida a codificación de entropía para reducir el ancho de banda requerido. Por consiguiente, los datos asociados con la información colateral son proporcionados al Codificador de entropía 122 según se muestra en la Figura 1A. Cuando se utiliza un modo de inter predicción, también hay que reconstruir una o varias imágenes de referencia en el extremo del codificador. En consecuencia, los residuos transformados y cuantificados son procesados por Cuantificación Inversa (IQ) 124 y Transformación Inversa (IT) 126 para recuperar los residuos. Los residuos se vuelven a añadir a los datos de predicción 136 en la Reconstrucción (REC) 128 para reconstruir los datos de vídeo. Los datos de vídeo reconstruidos pueden almacenarse en el búfer de imágenes de referencia 134 y utilizarse para la predicción de otros fotogramas.

Según se muestra en la Figura 1A, los datos de vídeo entrantes se someten a una serie de procesos en el sistema de codificación. Los datos de vídeo reconstruidos procedentes de REC 128 pueden ser sometidos a diversas degradaciones debidas a una serie de procesos. Por consiguiente, se aplican diversos procesos en bucle a los datos de vídeo reconstruidos antes de que los datos de vídeo reconstruidos se almacenen en el búfer de imágenes de referencia 134 con el fin de mejorar la calidad de vídeo. En el estándar de codificación de vídeo de alta eficiencia (HEVC) que está en desarrollo, se han desarrollado un Filtro de Desbloqueo (DF) 130, un Desplazamiento Adaptativo de Muestra (SAO) 131 y un Filtro de Bucle Adaptativo (ALF) 132 para mejorar la calidad de imagen. La información del filtro en bucle puede tener que ser incorporada en el flujo de bits para que un decodificador pueda recuperar adecuadamente la información requerida. Por lo tanto, se proporciona información de filtro en bucle de

SAO y ALF al Codificador de Entropía 122 para su incorporación en el flujo de bits. En la Figura 1A, se aplica primero el DF 130 al vídeo reconstruido; luego se aplica el SAO 131 al vídeo procesado por el DF; y se aplica el ALF 132 al vídeo procesado por el SAO. Sin embargo, el orden de procesamiento entre DF, SAO y ALF puede ser modificado.

En la Figura 1B se muestra un decodificador correspondiente al codificador de la Figura 1A. El flujo de bits de vídeo es decodificado por el Decodificador de Video 142 para recuperar los residuos transformados y cuantificados, información SAO/ALF y otra información del sistema. En el lado del decodificador, sólo se realiza la Compensación de Movimiento (MC) 113 en lugar de ME/MC. El proceso de decodificación es similar al del bucle de reconstrucción en el lado del codificador. Los residuos transformados y cuantificados, la información SAO/ALF y otra información del sistema recuperados se utilizan para reconstruir los datos de vídeo. El vídeo reconstruido es procesado posteriormente por el DF 130, SAO 131 y ALF 132 para producir el vídeo decodificado mejorado final.

HEVC ha adoptado ALF como un filtro en bucle sin desbloqueo para mejorar el rendimiento de la codificación. En la versión 5.0 del modelo de prueba de HEVC se describe un algoritmo de codificación ALF basado en imágenes. Sin embargo, normalmente se utiliza un esquema de codificación basado en LCU o un proceso de canalización (pipeline process) basado en LCU para implementaciones de codificadores y decodificadores de vídeo debido a un uso más eficiente de la memoria, un menor ancho de banda de memoria o un menor coste de hardware. Por lo tanto, el ALF basado en LCU es un enfoque preferido. Sin embargo, es deseable mejorar aún más el rendimiento del procesamiento ALF.

El SAO es otro procesamiento en bucle adoptado por HEVC para mejorar la calidad de imagen. El SAO consta de dos procedimientos. Uno es el desplazamiento de banda (BO), y el otro es el desplazamiento de borde (EO). BO se utiliza para clasificar los píxeles en múltiples bandas de acuerdo con la intensidad de píxel y se aplica un offset a los píxeles de cada banda. El EO se utiliza para clasificar los píxeles en categorías según las relaciones con los vecinos y se aplica un offset a los píxeles de cada categoría. En HM-5.0, una región puede seleccionar 7 tipos diferentes de SAO: 2 grupos de BO (grupo externo e interno), 4 patrones direccionales de EO (0°, 90°, 135° y 45°) y sin procesamiento (OFF). Además, una imagen puede ser dividida en múltiples regiones usando un procedimiento de partición de árbol cuádruple o ser dividida en regiones de unidades de codificación más grandes (LCU), y cada región tiene su propio tipo de SAO y valores de offset. Es deseable mejorar aún más el rendimiento del procesamiento SAO mejorando la señalización de parámetros SAO.

DESCRIPCIÓN RESUMIDA

Los objetivos antes mencionados son alcanzados. La invención tal como se define en el conjunto de reivindicaciones adjuntas; los demás ejemplos denominados formas de realización o cláusulas en la descripción son ejemplos ilustrativos, no formas de realización reivindicadas en la presente solicitud.

Se presentan un procedimiento y un aparato para decodificación de vídeo con un procesamiento en bucle de un vídeo reconstruido. Según una forma de realización útil para entender la presente invención, el procedimiento comprende recuperar datos de vídeo reconstruidos a partir de un flujo de bits de vídeo; recibir un indicador procedente del flujo de bits de vídeo; recibir información asociada con parámetros de filtro en bucle procedente de una carga útil de datos en el flujo de bits de vídeo para su compartición por dos o más bloques de codificación o datos de bloques de codificación individuales en el flujo de bits de vídeo de acuerdo con el indicador; y aplicar el procesamiento en bucle a bloques de codificación del vídeo reconstruido. El procesamiento en bucle puede corresponder a un Filtro de bucle adaptativo (ALF) o a un Desplazamiento adaptativo de muestra (SAO). El bloque de codificación puede corresponder a una unidad de codificación (CU), múltiples CU, una unidad de codificación más grande (LCU) o múltiples LCU. La carga útil de datos en el flujo de bits de vídeo se encuentra en un nivel de imagen, un conjunto de parámetros de adaptación (APS) o una cabecera de segmento. En una forma de realización útil para comprender la presente invención, el indicador es un indicador de intercalación que es utilizado para seleccionar si la información asociada con los parámetros del filtro en bucle está incorporada en la carga útil de datos en el flujo de bits de vídeo o intercalada con datos de bloques de codificación individuales en el flujo de bits de vídeo. En otra forma de realización útil para comprender la presente invención, el indicador está en un nivel de secuencia y la información asociada con los parámetros del filtro en bucle está incorporada en un Conjunto de parámetros de adaptación (APS), o una cabecera de segmento según el indicador.

Se divulgan un procedimiento y un aparato para codificación de vídeo con un procesamiento en bucle de un vídeo reconstruido. Según una forma de realización útil para comprender la presente invención, el procedimiento comprende recibir datos de vídeo reconstruidos; determinar parámetros de filtro en bucle asociados con el procesamiento en bucle, en el que el procesamiento en bucle se aplica a bloques de codificación del vídeo reconstruido; e incorporar información asociada con los parámetros de filtro en bucle en una carga útil de datos en un flujo de bits de vídeo para su compartición por dos o más bloques de codificación o intercalada con datos de bloques de codificación individuales en el flujo de bits de vídeo de acuerdo con un indicador. El procesamiento en bucle puede corresponder a un Filtro de bucle adaptativo (ALF) o a un Desplazamiento adaptativo de muestra

(SAO). El bloque de codificación puede corresponder a una unidad de codificación (CU), múltiples CU, una unidad de codificación más grande (LCU) o múltiples LCU. La carga útil de datos en el flujo de bits de vídeo se encuentra en un nivel de imagen, un conjunto de parámetros de adaptación (APS) o una cabecera de segmento. En una forma de realización útil para comprender la presente invención, el indicador es un indicador de intercalación que es utilizado para seleccionar si la información asociada con los parámetros del filtro en bucle es incorporada en la carga útil de datos en el flujo de bits de vídeo o intercalada con datos de bloques de codificación individuales en el flujo de bits de vídeo. En otra forma de realización útil para comprender la presente invención, el indicador está en un nivel de secuencia y la información asociada con los parámetros del filtro en bucle es incorporada en un Conjunto de parámetros de adaptación (APS), o una cabecera de segmento según el indicador.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

La Figura 1A ilustra un sistema de inter/intra codificación de vídeo adaptativa de ejemplo con un procesamiento en bucle DF, SAO y ALF.

La Figura 1B ilustra un sistema de inter/intra decodificación de vídeo adaptativa de ejemplo con un procesamiento en bucle DF, SAO y ALF.

La Figura 2 ilustra un ejemplo de uso compartido de parámetros de filtro en bucle entre un bloque actual y bloques vecinos.

La Figura 3 ilustra un ejemplo de clasificación mejorada de desplazamiento de borde basada en ocho píxeles vecinos en una ventana 3x3 alrededor del píxel actual.

La Figura 4 ilustra un diseño de sintaxis de ejemplo para seq_parameter_set_rbsp( ) que permite un control adaptativo de procesamiento SAO traspasando el límite de segmento.

La Figura 5 ilustra un diseño de sintaxis de ejemplo para aps_rbsp( ) que permite una señalización adaptativa de parámetros SAO para incorporar parámetros SAO en un APS o intercalarlos con datos de bloque en un segmento.

La Figura 6 ilustra un diseño de sintaxis de ejemplo para aps_sao_param( ) que permite que bloques de codificación en un segmento con aps_id referenciando al APS compartan parámetros SAO en el APS.

La Figura 7 ilustra un diseño de sintaxis de ejemplo para sao_unit_vlc() para incorporar información SAO para bloques de codificación.

La Figura 8 ilustra un diseño de sintaxis de ejemplo para sao_offset_vlc( ) para incorporar valores de offset SAO para bloques de codificación.

La Figura 9 ilustra un diseño de sintaxis de ejemplo para slice_header( ) para permitir la intercalación adaptativa de parámetros SAO con datos de bloques.

La Figura 10 ilustra un diseño de sintaxis de ejemplo para slice_data( ) para permitir la intercalación adaptativa de parámetros SAO con datos de bloques.

La Figura 11 ilustra un diseño de sintaxis de ejemplo para sao_unit_cabac() para permitir la intercalación adaptativa de parámetros SAO con datos de bloques.

La Figura 12 ilustra un diseño de sintaxis de ejemplo para sao_offset_cabac( ) para permitir la intercalación adaptativa de parámetros SAO con datos de bloques.

La Figura 13 ilustra un diseño de sintaxis de ejemplo para seq_parameter_set_rbsp( ) para permitir la señalización adaptativa de parámetros ALF.

La Figura 14 ilustra un diseño de sintaxis de ejemplo para slice_header( )para permitir la señalización adaptativa de parámetros ALF.

La Figura 15 ilustra un diseño de sintaxis de ejemplo para alf_param( ) para permitir la señalización adaptativa de parámetros ALF.

La Figura 16 ilustra un diseño de sintaxis de ejemplo para alf_unit( ) para permitir la señalización adaptativa de parámetros ALF.

La Figura 17 ilustra un diseño de sintaxis de ejemplo para alf_info( ) para permitir la señalización adaptativa de parámetros ALF.

DESCRIPCIÓN DETALLADA

En el filtrado en bucle basado en bloques, los parámetros del filtro en bucle para cada bloque tienen que ser suministrados al lado del decodificador para una operación apropiada de filtrado en bucle en el lado del decodificador. Ejemplos de filtros en bucle pueden incluir el desplazamiento adaptativo de muestra (SAO) o el filtro de bucle adaptativo (ALF). El filtro en bucle también se denomina procesamiento de filtro en bucle o procesamiento en bucle en esta divulgación. Los parámetros en bucle también se denominan información en bucle en esta divulgación. Cuando el tamaño de bloque es pequeño, tal como una unidad de codificación (CU) o una CU más grande (LCU), la tasa de bits correspondiente a los parámetros del filtro en bucle se hace relativamente grande. Por lo tanto, es deseable reducir la tasa de bits asociada con los parámetros del filtro en bucle. Una forma de realización según la presente invención utiliza un indicador de fusión para indicar el caso en que un bloque actual comparte parámetros de filtro en bucle con un bloque vecino o bloques vecinos. Además, el bloque puede incluir múltiples LCU para reducir la tasa de bits correspondiente a los parámetros del filtro en bucle. El indicador de fusión también se puede determinar implícitamente. Por ejemplo, si los bloques a la izquierda y superiores tienen los mismos parámetros de filtro en bucle, el bloque actual compartirá los parámetros de filtro en bucle con sus vecinos izquierdo

y superior sin necesidad de transmitir explícitamente el indicador de fusión. La información del filtro en bucle puede incorporarse al flujo de bits de vídeo comprimido.

La compartición de parámetros de filtro en bucle para un bloque actual puede seleccionar uno de los bloques vecinos, tal como el bloque izquierdo, superior, derecho, superior izquierdo, inferior, inferior izquierdo, etcétera. La sintaxis de fusión se utiliza para indicar que dos bloques se fusionan entre sí y comparten los mismos parámetros de filtro en bucle. La Figura 2 ilustra un ejemplo de compartición de parámetros de filtro en bucle con un bloque vecino. A continuación se describe un proceso de ejemplo para determinar si un bloque C actual debe compartir parámetros de filtro en bucle con un bloque vecino. Si la información SAO del bloque A no es igual a la del bloque B, X1 se establece igual a uno. De lo contrario, X1 se establece igual a cero. Si la información SAO del bloque D no es igual a la del B, X2 se establece igual a uno. De lo contrario, X2 se establece igual a cero. Si la información SAO del bloque D no es igual a la del bloque A, X3 se establece igual a uno. De lo contrario, X3 se establece igual a cero. La variable X puede calcularse según X = X3*4 + X2*2 + X1*1. La variable X se puede utilizar para seleccionar un bloque vecino para su fusión con el bloque actual.

En un proceso de fusión de ejemplo, cuando X es igual a 7, se incorpora un indicador de fusión para indicar si el bloque C utiliza nuevos parámetros de filtro en bucle o no. Cuando el indicador de fusión tiene un valor igual a uno, el bloque C utiliza nuevos parámetros de filtro en bucle y, en caso contrario, el bloque C comparte parámetros de filtro en bucle con el bloque A o B. Cuando X es igual a 5, se incorpora un indicador de fusión para indicar si el bloque C utiliza nuevos parámetros de filtro en bucle o no. Cuando el indicador de fusión tiene un valor igual a uno, el bloque C utiliza nuevos parámetros de filtro en bucle y, en caso contrario, el bloque C comparte parámetros de filtro en bucle con el bloque A. Cuando X es igual a 3, se incorpora un indicador de fusión para indicar si el bloque C utiliza nuevos parámetros de filtro en bucle o no. Cuando el indicador de fusión tiene un valor igual a uno, el bloque C utiliza nuevos parámetros de filtro en bucle y, en caso contrario, el bloque C comparte parámetros de filtro en bucle con el bloque B. Cada bloque puede ser una CU, múltiples CU, una LCU, múltiples LCU u otra estructura de bloques. El bloque también puede corresponder a diferentes tamaños. El bloque también se denomina bloque de codificación en esta divulgación.

En un enfoque convencional, el procesamiento SAO sólo se aplica al componente de luma. Una forma de realización según la presente invención también aplica de forma selectiva el procesamiento SAO a los componentes de croma si el procesamiento SAO es aplicado al componente de luma. Cuando el procesamiento SAO se aplica al componente de luma, se puede utilizar un indicador SAO de croma para indicar si el procesamiento SAO se aplica a los componentes de croma. La información SAO para un componente de croma puede derivarse a partir de la información SAO para el componente de luma u otro(s) componente(s) de croma. Cuando un componente de croma comparte la información SAO con el componente de luma u otro(s) componente(s) de croma, se puede utilizar un indicador de compartición de información SAO para indicar el caso. La información SAO puede incluir valores de offset, tipo de SAO (también denominado categoría o clase en esta divulgación), y decisión ON/OFF. En una forma de realización, los valores de offset son predefinidos de modo que se puede usar un índice para seleccionar uno de los valores de offset. La información SAO puede ser incorporada en una carga útil de datos en el flujo de bits tal como PPS, APS, o una cabecera de segmento para compartir múltiples bloques.

La información asociada con parámetros SAO puede codificarse en base a parámetros SAO de un bloque procesado anteriormente utilizando un procedimiento predictivo. La codificación del valor de offset puede depender del aumento interno de la profundidad de bits, información de modo inter/intra, información de movimiento, tamaño de transformación, tamaño de etapa de cuantificación, residual, información de desbloqueo, tamaño de imagen y tamaño de región. Por ejemplo, si el aumento interno de la profundidad de bits es mayor que uno, la codificación del valor de offset puede incrementarse en un bit. En otro ejemplo de codificación de información SAO, los parámetros SAO pueden cambiar su precisión de codificación de acuerdo con los parámetros de cuantificación. Con el fin de reducir la tasa de bits asociada con los parámetros SAO, puede codificarse la información de valores de offset de forma retardada en el tiempo. Además, la información SAO de un segmento actual puede ser compartida con otros segmentos o regiones. Por consiguiente, puede incorporarse la información de offset en el conjunto de parámetros de imagen (PPS). Los bloques de codificación de un segmento o región actual pueden compartir la misma información SAO.

En un enfoque convencional, el procesamiento SAO se aplica a menudo después del procesamiento DF. Una forma de realización según la presente invención puede aplicarse de forma selectiva entre el procesamiento DF y el procesamiento SAO a un bloque. Alternativamente, tanto el procesamiento DF como el procesamiento SAO pueden aplicarse a los mismos datos de vídeo reconstruidos y combinarse linealmente los outputs de ambos procesos.

De acuerdo con una forma de realización de la presente invención, el procedimiento de clasificación de píxeles para SAO puede combinarse con otros procedimientos de clasificación de píxeles tales como dirección de borde, intensidad de píxeles, variación de píxeles, varianza de píxeles, suma de Laplace de píxeles, resultado de filtrado de paso alto, resultado de filtrado de paso bajo, valor absoluto del resultado de filtrado de paso alto y valor medio de

píxeles vecinos. Por ejemplo, una categoría de Desplazamiento de banda (BO) o Desplazamiento de borde (EO) puede dividirse aún más por EO. En otro ejemplo, una categoría de EO puede ser dividida por BO o suma de Laplace.

En la clasificación convencional de EO, se utilizan dos píxeles vecinos en una ventana 3x3 para clasificar un píxel actual en diferentes categorías o clases. Una forma de realización según la presente invención puede utilizar un procedimiento de clasificación mejorado basado en todos los píxeles vecinos en la ventana 3x3. Los píxeles vecinos (P1-P8) alrededor del píxel actual C se muestran en la Figura 3. Como ejemplo, se puede definir un índice de clase ClassIdx de la siguiente manera:

$$ClassIdx = Index2ClassTable(f(C,P_1) + f(C,P_2) + ... + f(C,P_8)),$$

en el que $f(C,P_n)$ es una función de comparación e Index2ClassTable es una función de mapeo para mapear los resultados de la comparación con el índice de clase. La función de comparación f(x,y) se define de la siguiente manera:

$$\begin{aligned} &si\ x\text{-}y >= th, && f(x,y) = 1, \\ &si\ x=y, && f(x,y) =0, \quad y \\ &si\ x\text{-}y<th, && f(x,y) =-1 \end{aligned}$$

en el que th es un umbral.

Una función de comparación alternativa f(x,y) puede definirse de la siguiente manera:

$$\begin{aligned} &si\ (x/s)\text{-}(y/s) >= th, && f(x,y) = 1, \\ &si\ (x/s)=(y/s), && f(x,y) =0, \quad y \\ &si\ (x/s)\text{-}(y/s)<th, && f(x,y) =-1. \end{aligned}$$

en el que th es un umbral y s es un factor de escala.

El procedimiento mejorado de clasificación EO puede aplicarse al procedimiento de clasificación combinada de píxeles de SAO y a otros procedimientos de clasificación de píxeles. Por ejemplo, una categoría de desplazamiento de borde (EO) de acuerdo con la clasificación EO mejorada mencionada anteriormente puede ser dividida por EO o BO.

En un enfoque convencional, la unidad utilizada para la codificación ALF basada en LCU es siempre una LCU. La mejora en distorsión debida al ALF está relacionada con el tamaño de la LCU. Una LCU más pequeña suele permitir que el diseño del ALF se adapte mejor a las características locales. Sin embargo, la cantidad de información asociada con los parámetros ALF es relativamente constante e independiente del tamaño de LCU. Por lo tanto, un tamaño de LCU más pequeño resultará en una tasa de bits más alta asociada con la información correspondiente a los parámetros ALF. Por consiguiente, la tasa de bits neta disponible para codificar datos de vídeo puede reducirse considerablemente y puede degradarse el rendimiento del sistema. Con el fin de superar este problema, una forma de realización según la presente invención agrupa múltiples LCU en una unidad, denominada unidad de filtro o bloque de codificación en esta divulgación. Por consiguiente, se aplica el mismo ALF a todas las LCU de una unidad de filtro y los parámetros ALF son compartidos entre todas las LCU de una unidad de filtro para reducir la tasa de bits necesaria para incorporar los parámetros ALF. El conjunto de parámetros ALF puede incluir uno o más elementos seleccionados de un conjunto que consta de coeficientes de filtro, forma de filtro, tamaño de filtro, control ON/OFF e información de región. Se puede utilizar un búfer para almacenar el conjunto de parámetros ALF/SAO de modo que la información pueda ser compartida por otra unidad de filtro, segmento o imagen. La unidad de filtro puede ser tan grande como una imagen o múltiples LCU. Por ejemplo, una unidad de filtro puede consistir en LCU de MxN, en el que M y N son enteros mayores que cero. Los límites de una unidad de filtro pueden estar o no estar alineados con los límites de LCU. Cuando se utiliza una unidad de filtro, los parámetros ALF se pueden diseñar en base a las estadísticas asociadas con la unidad de filtro. Los parámetros ALF diseñados pueden aplicarse a todos los píxeles en la unidad de filtro. El codificador puede determinar el valor de M y N e incorporar la información del tamaño del filtro en una capa de secuencia RBSP (carga útil de secuencia de bytes sin procesar) o en una capa de imagen RBSP. Por lo tanto, puede reducirse la información colateral correspondiente a ALF compartiendo los parámetros en bucle entre múltiples LCU.

Los candidatos a filtro para una unidad de filtro pueden ser derivados en base a las unidades de filtro subyacentes, o los candidatos a filtro pueden compartir al menos una parte de los candidatos a filtro usados por unidades de filtro procesadas anteriormente en el segmento actual. Sin embargo, para la primera unidad de filtro en un segmento, no hay una unidad de filtro procesada anteriormente en el segmento para que la comparta la unidad de filtro actual. Por lo tanto, tienen que utilizarse candidatos a filtro por defecto u otros medios para procesar la primera unidad de filtro y puede degradarse el rendimiento. En un enfoque convencional, los parámetros ALF se derivan a partir de una unidad de filtro (una LCU o una imagen) sin ninguna información procedente de otras unidades de filtro. Una forma de realización según la presente invención permite utilizar la información procedente de un fotograma anterior o unidades de filtro procesadas anteriormente para derivar algunos candidatos a filtro para una unidad de filtro actual. Por ejemplo, se pueden utilizar las estadísticas de unidades de filtro con el modo ALF-OFF en el fotograma anterior para derivar parámetros de un filtro y se puede utilizar el filtro como un candidato a filtro para la unidad de filtro actual. El filtro derivado a partir de las unidades de filtro ALF-ON en el fotograma anterior puede ser utilizado como otro candidato a filtro. Además, una imagen se puede dividir en varias particiones y se pueden derivar los respectivos filtros para las unidades de filtro ALF-ON y ALF-OFF en cada partición. Uno de estos filtros puede ser utilizado como candidato a filtro para las unidades de filtro actuales. Las estadísticas de unidades procesadas anteriormente en un segmento actual pueden acumularse para derivar candidatos a filtro en base a las estadísticas acumuladas.

De acuerdo con el procesamiento ALF convencional, el procesamiento de componentes de croma puede ser independiente del procesamiento de componentes de luma o siempre sigue el procesamiento de componentes de luma. Una forma de realización según la presente invención combina ambos procedimientos de forma adaptativa. Se utiliza una sintaxis para indicar si el componente de croma es filtrado o no, y se utiliza otra sintaxis para indicar si los componentes de croma comparten el filtro con el componente de luma o si utiliza el suyo propio como incorporado en el flujo de bits. Por lo tanto, los coeficientes de filtro para los componentes de croma pueden derivarse a partir del filtro de luma o decodificarse a partir de la corriente de bits. Además, con el fin de reducir la información colateral asociada con los parámetros ALF, la huella de filtro para los componentes de croma puede ser un subconjunto de los filtros para el componente de luma.

Cuando se comparten parámetros ALF anteriores, los parámetros en unidades de filtro anteriores del segmento actual pueden ser reutilizados por las unidades de filtro posteriores. Para el procesamiento de SAO, los parámetros en regiones o LCU anteriores del segmento actual pueden ser reutilizados por las siguientes regiones o LCU. Para mayor comodidad, la unidad de filtro también puede referirse a una región o a una LCU. La magnitud de cuanto más atrás se pueden reutilizar los parámetros ALF anteriores puede ser definido por un usuario o puede depender del tamaño de imagen. Cuando se comparte la información ALF/SAO, la información ALF/SAO puede derivarse o copiarse a partir de una región codificada anteriormente, de una imagen codificada anteriormente o de información ALF/SAO predefinida. La información ALF puede incluir uno o más elementos del conjunto que consiste en coeficientes de filtro, forma de filtro, tamaño de filtro, control ON/OFF e información de región.

Cada unidad de filtro puede utilizar un índice para seleccionar parámetros ALF/SAO anteriores almacenados en un búfer. El índice se puede codificar por entropía de forma adaptativa para reducir la tasa de bits. Por ejemplo, a un conjunto de parámetros ALF/SAO seleccionados con mayor frecuencia se le puede asignar una palabra clave más corta. En otro ejemplo, se puede utilizar una técnica de codificación predictiva en la que se determinan uno o más modos más probables. Si el índice actual es igual a uno de los modos más probables, se puede utilizar una palabra clave muy corta para codificar el índice. De lo contrario, necesitará más bits para identificar cuál de los índices restantes es el mismo que el índice actual. La técnica es similar a la técnica de codificación de modo más probable utilizada en la codificación modo intra. Se puede incorporar la correspondiente sintaxis para el índice en el conjunto de parámetros de adaptación (APS), conjunto de parámetros de imagen (PPS), una cabecera de segmento o datos de segmento. Puede incorporarse información asociada con ALF en el conjunto de parámetros de adaptación (APS), una cabecera de segmento, datos de segmento o ser modificada de forma adaptativa en base a un indicador incorporado en el conjunto de parámetros de secuencia (SPS), el conjunto de parámetros de imagen (PPS), el conjunto de parámetros de adaptación (APS) o una cabecera de segmento de acuerdo con una forma de realización de la presente invención.

En otro ejemplo para habilitar la señalización adaptativa de parámetros ALF, se puede utilizar un indicador de intercalación de parámetros ALF o un indicador de intercalación de parámetros SAO para indicar si los parámetros ALF para cada unidad de filtro (como se ha mencionado anteriormente, la unidad de filtro puede referirse a una región de una LCU) son intercalados con datos de unidad de filtro en el segmento. El indicador de intercalación de parámetros ALF o el indicador de intercalación de parámetros SAO pueden incorporarse en el conjunto de parámetros de imagen (PPS), el conjunto de parámetros de adaptación (APS) o una cabecera de segmento. Además, el indicador para indicar la intercalación puede incorporarse en múltiples RBSP simultáneamente, tales como un APS y una cabecera de segmento. Cuando el indicador para indicar la intercalación se encuentra en múltiples RBSP, el indicador debe tener el mismo valor en los múltiples RBSP. Por consiguiente, la forma de

realización de la presente invención también proporciona la capacidad de redundancia para proteger el indicador de intercalación.

Para SAO, se puede utilizar un indicador, sao_max_region_size_minus_one, para indicar el número máximo de LCU en una región de procesamiento SAO. El número máximo de LCU para una región de procesamiento SAO es sao_max_region_size_minus_one + 1. El indicador, sao_max_region_size_minus_one puede incorporarse en un conjunto de parámetros de imagen (PPS), conjunto de parámetros de secuencia (SPS), conjunto de parámetros de adaptación (APS), cabecera de segmento, o más de uno de los anteriores. Las LCU en la misma región de procesamiento SAO pueden compartir los mismos parámetros SAO. Por ejemplo, si sao_max_region_size_minus_one es igual a cero, el número máximo de LCU en una región de procesamiento SAO se establece igual a uno. Si se utilizan el diseño de sintaxis basado en LCU y la codificación “series” ("run"), el valor “series” ("run") indica el número de LCU que comparten los mismos parámetros SAO. Para algunas aplicaciones, se puede utilizar un tamaño de LCU pequeño, tal como 16x16. En este caso, la tasa de bits asociada con la codificación por longitud de series de LCU consecutivas en una región que comparte parámetros SAO puede ser relativamente alta. Por lo tanto, el valor “series” ("run") se utiliza para indicar el número de regiones que comparten los mismos parámetros SAO.

La codificación por longitud de series (run-length encoding) también se puede aplicar a los parámetros ALF. Una unidad de filtro puede consistir en una LCU y el procesamiento basado en unidad de filtro pasa a ser un procesamiento basado en LCU en este caso. Unidades de filtro consecutivas pueden compartir los mismos parámetros ALF. Para reducir la tasa de bits para indicar la compartición de parámetros ALF, se utiliza la codificación por longitud de series para indicar el número de unidades de filtro consecutivas que comparten los parámetros ALF con la unidad de filtro actual.

Para el caso de segmentos de granularidad fina, el bloque de granularidad puede ser más pequeño que una LCU. En este caso, una LCU puede incluir más de un dato de segmento, es decir, una LCU puede contener más de un conjunto de parámetros ALF/SAO. Una forma de realización según la presente invención forzará que todos los parámetros de filtro en bucle en una LCU sean los mismos.

El diseño en bucle basado en imágenes tiene que esperar hasta que la imagen completa esté disponible. Puede causar un retraso en el procesamiento. Sin embargo, puede que no sea un problema para las imágenes que no son de referencia. Una forma de realización según la presente invención puede simplemente saltar la etapa de filtrado real después de que se hayan derivado los parámetros de filtro. En cambio, en un enfoque basado en imágenes, el procesamiento en bucle se puede aplicar a imágenes que no son de referencia sin causar ninguna latencia de codificación adicional o acceso a memoria de imágenes adicional.

El procesamiento de filtro en bucle en el lado del codificador implica dos etapas separadas. En la primera etapa, se recogen las estadísticas a partir de la LCU o imagen subyacente. Los parámetros del filtro en bucle se derivan en base a las estadísticas recopiladas. En la segunda etapa, se aplica el procesamiento del filtro en bucle a los píxeles de la LCU o imagen en base a los parámetros del filtro en bucle derivados. Dado que cada procesamiento en bucle, tal como el procesamiento ALF, se realiza en etapas separadas, puede causar un acceso a datos considerable. Una forma de realización según la presente invención combina una de las etapas del procesamiento de filtro en bucle con otro procesamiento de filtro en bucle para reducir el acceso a datos asociado. Por ejemplo, la primera etapa del proceso ALF/SAO es recopilar estadísticas, lo que se puede realizar junto con otras herramientas de codificación basadas en LCU, tal como el proceso de desbloqueo. Por consiguiente, las estadísticas para el procesamiento ALF/SAO pueden ser recopiladas durante el proceso de desbloqueo. Por lo tanto, los parámetros ALF/SAO pueden ser derivados sin necesidad de acceso adicional a la memoria de imágenes. La segunda etapa del proceso ALF/SAO implica aplicar un filtro a datos de píxel, lo que se puede realizar durante el proceso de estimación de movimiento posterior. Por lo tanto, el proceso ALF/SAO en el lado del codificador puede realizarse sin ningún acceso a memoria separado, lo que se denomina codificación de cero pasos en esta divulgación. En el lado del decodificador, no hay necesidad de recopilar estadísticas para el diseño del filtro ALF/SAO. Sin embargo, el decodificador todavía puede aprovechar la decodificación de cero pasos realizando el procesamiento de filtro ALF/SAO durante la compensación de movimiento.

Para algunas aplicaciones de latencia baja, se prefiere el procesamiento basado en unidades de filtro. Para la codificación basada en unidad de filtro, se espera que la decisión de control ON/OFF de una unidad de filtro finalice una vez que se disponga de los resultados de la codificación de unidad de filtro. Además, es preferible que los datos comprimidos asociados con la unidad de filtro sean intercalados con los parámetros de filtro en bucle en un segmento. Para aplicaciones de latencia baja, el control ON/OFF a nivel de segmento puede causar una gran latencia de codificación. Por consiguiente, una forma de realización de la presente invención siempre establece explícitamente el indicador de control ON/OFF a nivel de segmento en ON cuando los parámetros de filtro en bucle son codificados e intercalados con datos de unidad de filtro (también denominados datos de bloque en esta divulgación) en un segmento. Alternativamente, una forma de realización de la presente invención puede deshabilitar

condicionalmente el indicador de control ON/OFF a nivel de segmento. Si los parámetros de filtro en bucle son codificados e intercalados en un segmento, no se envía el indicador de control ON/OFF a nivel de segmento. De lo contrario, se envía el indicador de control ON/OFF a nivel de segmento.

En un enfoque convencional, el número de desplazamientos (offset) en cada región para BO y EO es diferente. Una forma de realización según la presente invención unifica el número de desplazamientos en cada región para BO y EO. En consecuencia, el número de grupos de BO es cambiado a ocho, y cada grupo de BO tiene cuatro desplazamientos para cada región. Reduciendo el número de desplazamientos para cada grupo puede reducir la tasa de bits asociada. La tasa de bits asociada con los desplazamientos también puede ser reducida según una forma de realización de la presente invención. Por ejemplo, el rango de desplazamientos puede limitarse a un rango más pequeño. Esto será útil para regiones pequeñas en las que se espera que los desplazamientos sean menores. El búfer necesario para la predicción de desplazamientos basada en bloques vecinos puede ser reducido según una forma de realización de la presente invención. Por ejemplo, la predicción de desplazamientos puede evitar usar el offset para la LCU por encima de la LCU actual. Las operaciones de filtrado en bucle incluyen filtro de desbloqueo, SAO y ALF. El procesamiento SAO de croma puede ser mejorado según una forma de realización de la presente invención habilitando condicionalmente el procesamiento SAO de croma dependiendo de si el procesamiento SAO de luma está habilitado. En otra forma de realización según la presente invención, los desplazamientos para componentes de croma y luma pueden ser compartidos cuando se selecciona el EO.

Se ilustran diversos diseños de sintaxis en el presente documento como formas de realización de ejemplo de acuerdo con la presente invención. La Figura 4 ilustra una estructura de sintaxis de ejemplo del Conjunto de Parámetros de Secuencia de acuerdo con la presente invención, en el que se incorpora un indicador loop_filter_across_slice_flag. Si el indicador loop_filter_across_slice_flag es igual a uno, las operaciones de filtrado en bucle se pueden realizar traspasando el límite del segmento. Si el indicador Loop_filter_across_slice_flag es igual a cero, las operaciones de filtrado en bucle son independientes de segmento y las operaciones no traspasan los límites de segmento. Por consiguiente, el búfer requerido para una predicción de parámetros de filtro en bucle basada en bloques vecinos puede ser reducido según una forma de realización de la presente invención.

La Figura 5 ilustra un ejemplo de incorporación de un indicador de intercalación de parámetros SAO en el APS. En la Figura 5, el aps_id identifica el conjunto de parámetros de adaptación (APS) que es referenciado por bloques de codificación en un segmento con el correspondiente aps_id en la cabecera del segmento. Cuando el indicador de intercalación de parámetros SAO, es decir, el aps_sao_interleaving_flag es igual a uno, los parámetros SAO son intercalados con los datos de unidad de filtro para los segmentos que referencian al APS, según indica el aps_id. Cuando el indicador aps_sao_interleaving_flag es igual a cero, los parámetros SAO son incorporados en el APS para los segmentos que referencian al APS. Otro indicador, es decir, aps_sample_adaptive_offset_flag, es incorporado en el APS para controlar la activación/desactivación (ON/OFF) del filtro. Si el indicador aps_sample_adaptive_offset_flag es igual a uno, el SAO está en ON para los segmentos que referencian al APS según indica el aps_id. Por otro lado, si el indicador aps_sample_adaptive_offset_flag es igual a cero, el SAO está en OFF para los segmentos que referencian al APS según indica el aps_id.

La Figura 6 ilustra un diseño de sintaxis de ejemplo, aps_sao_param( ), para parámetros SAO en el APS según una forma de realización de la presente invención. La estructura de sintaxis incluye la información SAO necesaria para el procesamiento SAO. Por ejemplo, la estructura de sintaxis puede incluir un indicador o indicadores, tales como sao_cb_enable_flag y sao_cr_enable_flag en la Figura 6 para determinar si el procesamiento SAO se aplica a respectivos componentes de croma de la imagen actual. La estructura de sintaxis también puede incluir información sobre el tamaño de imagen en términos de tamaño de LCU, tales como sao_num_lcu_in_width_minus1 y sao_num_lcu_in_height_minus1 en la Figura 6. La estructura de sintaxis también puede incluir información sobre si todas las unidades de codificación de un segmento se procesan utilizando los mismos parámetros SAO o parámetros SAO individuales para cada unidad de codificación más grande o unidad de filtro, según indican en la Figura 6 los indicadores sao_one_luma_unit_flag, sao_one_cb_unit_flag y sao_one_cr_unit_flag respectivamente. Si alguno de los indicadores anteriores tiene un valor igual a uno, se incorporarán valores de offset de SAO en sao_offset_vlc() para los bloques de codificación en el segmento a compartir. La estructura de sintaxis también puede incluir una indicación, tal como sao_repeat_row_flag[cIdx], sobre si los parámetros SAO de los bloques de codificación en la fila de bloques de codificación actual, tal como una fila de LCU, son los mismos que los del bloque de codificación superior para el índice de componente de color respectivo, cIdx.

En la estructura de sintaxis de ejemplo anterior, aps_sao_param( ), también se incorpora una estructura de bloques SAO, sao_unit_vlc() junto con un respectivo indicador de repetición de fila. La Figura 7 ilustra una estructura de sintaxis de ejemplo, sao_unit_vlc(), que incluye información asociada con información de compartición de parámetros SAO entre bloques de codificación, tales como sao_run_diff y sao_merge_up_flag. El número de veces que los parámetros SAO correspondientes a un bloque de codificación se repiten para bloques de codificación subsiguientes en la misma fila es representado por saoRun[cIdx][rx][ry]. El índice de matriz cIdx especifica el componente de color; cIdx tiene un valor igual a 0, 1 o 2 correspondiente a luma, Cb o Cr respectivamente. Los

índices rx y ry especifican la ubicación del bloque de codificación subyacente en relación con el bloque de codificación superior izquierdo de la imagen. El elemento de sintaxis, sao_run_diff especifica el saoRun[ ][ ][ ][ ] del bloque de codificación actual si la fila actual es la primera fila, o bien especifica la diferencia entre la serie (run) del bloque de codificación actual y la serie (run) del bloque de codificación superior. Cuando saoRun[ ][ ][ ][ ] es mayor o igual que cero, los elementos de sintaxis sao_offset_vlc() se derivan a partir de correspondientes elementos de sintaxis del bloque de codificación izquierdo. La longitud del elemento de sintaxis sao_run_diff es de Ceil(Log2(sao_num_lcu_in_width_minus 1 – rx + 2)) bits. Cuando el indicador, sao_merge_up_flag es igual a uno, los elementos de sintaxis de sao_offset_vlc() se derivan a partir de correspondientes elementos de sintaxis del bloque de codificación superior. Cuando el indicador, sao_merge_up_flag es igual a cero, los elementos de sintaxis de sao_offset_vlc() no se derivan a partir de correspondientes elementos de sintaxis del bloque de codificación superior. Cuando sao_merge_up_flag no está presente, se infiere que es igual a cero.

La Figura 8 ilustra una estructura de sintaxis de ejemplo para sao_offset_vlc() según una forma de realización de la presente invención. El elemento de sintaxis sao_type_idx[cIdx][rx][ry] indica el tipo de SAO que puede ser BO (Desplazamiento de Banda) o EO (Desplazamiento de borde). Cuando sao_type_idx[cIdx][rx][ry] tiene un valor igual a 0, indica que el SAO está en OFF; un valor igual a uno hasta cuatro, indica que se utiliza una de las cuatro categorías de EO correspondientes a 0°, 90°, 135° y 45°; y un valor igual a cinco, indica que se utiliza BO. En el ejemplo anterior, tanto el tipo BO como EO tienen cuatro valores de offset SAO.

La Figura 9 ilustra una estructura de sintaxis de ejemplo para una cabecera de segmento para permitir que los parámetros SAO se incorporen en una cabecera de segmento adaptativamente de acuerdo con una forma de realización de la presente invención. Cuando SAO está habilitado, entre otras condiciones, según lo indicado por el indicador, sample_adaptive_offset_enabled_flag, se incorporan dos indicadores adicionales, en concreto slice_sao_interleaving_flag y slice_sample_adaptive_offset_flag en la cabecera del segmento. Si slice_sao_interleaving_flag es igual a uno, los parámetros SAO son intercalados con los datos de unidad de filtro en los datos de segmento. Si slice_sao_interleaving_flag es igual a cero, los parámetros SAO utilizan información incorporada en el APS al que referencia el aps_id. El indicador para indicar intercalación puede incorporarse en múltiples RBSP simultáneamente, tal como el APS y la cabecera del segmento. Cuando el indicador para indicar la intercalación existe en múltiples RBSP, el indicador debe tener el mismo valor en los múltiples RBSP. Por consiguiente, cuando hay un APS activo, el valor de slice_sao_interleaving_flag en la cabecera del segmento será el mismo que el de aps_sao_interleaving_flag en el APS. Si slice_sample_adaptive_offset_flag es igual a uno, el SAO está en ON (activado) para el segmento actual. Si slice_sample_adaptive_offset_flag es igual a cero, el SAO está en OFF (desactivado) para el segmento actual. De forma similar, cuando hay un APS activo, el valor de slice_sample_adaptive_offset_flag será el mismo que el de aps_sample_adaptive_offset_flag. Por consiguiente, la forma de realización de la presente invención también proporciona la capacidad de redundancia para proteger el indicador de intercalación.

La Figura 10 ilustra una estructura de sintaxis de ejemplo, slice_data( ), para que los datos de segmento admitan la señalización adaptativa de parámetros SAO de acuerdo con una forma de realización de la presente invención. Según se muestra en la Figura 10, cuando slice_sao_interleaving_flag tiene un valor igual a uno, los datos individuales de unidad SAO (es decir, sao_unit_cabac ( )) son incorporados si el correspondiente indicador de habilitación de SAO está activado.

La Figura 11 ilustra una estructura de sintaxis de ejemplo, sao_unit_cabac (), para datos de unidad SAO de acuerdo con una forma de realización de la presente invención. Unos indicadores de fusión, sao_merge_left_flag y sao_merge_up_flag se utilizan para indicar si el bloque de codificación actual comparte desplazamientos (offset) SAO con la unidad de codificación izquierda o superior respectivamente. Cuando la unidad de codificación actual no comparte parámetros SAO con su bloque vecino de la izquierda o superior, se incorporan desplazamientos (offset) SAO, sao_offset cabac(), para el bloque de codificación actual.

La Figura 12 ilustra una estructura de sintaxis de ejemplo para sao_offset_cabac( ) según una forma de realización de la presente invención. El elemento de sintaxis sao_type_idx[cIdx][rx][ry] indica el tipo de SAO que puede ser BO (Desplazamiento de Banda) o EO (Desplazamiento de borde). Cuando sao_type_idx[cIdx][rx][ry] tiene un valor igual a 0, indica que el SAO está en OFF; un valor igual a uno hasta cuatro, indica que se utiliza uno de los cuatro OE correspondientes a 0°, 90°, 135° y 45°; y un valor igual a cinco, indica que se utiliza BO. En el ejemplo anterior, tanto el tipo BO como el tipo EO tienen cuatro valores de offset SAO. Cuando el sao_type_idx[cIdx][rx][ry] no está presente, se puede inferir el sao_type_idx[cIdx][rx][ry]. Por ejemplo, si sao_merge_up_flag es igual a uno, el sao_type_idx[cIdx][rx][ry] se establece igual al sao_type_idx[cIdx][rx][ry-1]. De lo contrario, el sao_type_idx[cIdx][rx][ry] se establece para que sea igual al sao_type_idx[cIdx][rx-1][ry].

La Figura 13 ilustra un ejemplo de diseño de sintaxis seq_parameter_set_rbsp(), para que el SPS admita un indicador para incorporar información ALF de forma adaptativa. Según se muestra en la Figura 13, el flag, adaptive_loop_filter_enabled_flag se utiliza para indicar si se permite la incorporación de parámetros ALF de forma

adaptativa. Cuando se habilita la incorporación de parámetros ALF de forma adaptativa según lo indicado por el adaptive_loop_filter_enabled_flag, se utiliza otro indicador, alf_coef_in_slice_flag, para indicar dónde se incorporan los parámetros ALF. Cuando alf_coef_in_slice_flag es igual a uno, se incorpora la sintaxis alf_param() para parámetros ALF en la cabecera de segmento. Cuando alf_coef_in_slice_flag es igual a cero, la sintaxis alf_param() para parámetros ALF será incorporada en el APS. En una sintaxis a nivel de segmento, si alf_coef_in_slice_flag es igual a uno, los parámetros ALF pueden ser incorporados en una cabecera de segmento. Además, los parámetros de control ON/OFF de CU de ALF no serán incorporados al nivel de segmento.

La Figura 14 ilustra un diseño de cabecera de segmento de ejemplo que permite incorporar parámetros ALF en la cabecera de segmento de forma adaptativa de acuerdo con los indicadores adaptive_loop_filter_enabled_flag y alf_coef_in_slice_flag mencionados en la Figura 13. Cuando ALF está habilitado, según lo indicado por el indicador adaptive_loop_filter_enabled_flag, se utiliza otro indicador, slice_adaptive_loop_filter_flag, para indicar si se aplica ALF a nivel de segmento. Si se aplica ALF a nivel de segmento y alf_coef_in_slice_flag indica que los parámetros ALF son incorporados en la cabecera del segmento, se incorpora la sintaxis alf_param() en la cabecera del segmento. Por otro lado, si se aplica ALF a nivel de segmento y alf_coef_in_slice_flag indica que los parámetros ALF no son incorporados en la cabecera del segmento, se incorporará la sintaxis alf_cu_control_param() en la cabecera del segmento.

La Figura 15 ilustra un diseño de sintaxis de ejemplo para parámetros ALF de acuerdo con una forma de realización de la presente invención. La estructura de sintaxis contiene la información ALF necesaria para el procesamiento ALF. Por ejemplo, la estructura de sintaxis puede incluir uno o varios indicadores, tal como alf_cb_enable_flag y alf_cr_enable_flag de la Figura 15 para determinar si se aplica el procesamiento ALF a respectivos componentes de croma de la imagen actual. La estructura de sintaxis también puede incluir información sobre el tamaño de imagen en términos de tamaño de LCU, tal como alf_num_lcu_in_width minus1 y alf_num_lcu_in_height minus1 de la Figura 15. La estructura de sintaxis también puede incluir información sobre si todas las unidades de codificación de un segmento se procesan utilizando los mismos parámetros ALF o parámetros ALF individuales para cada unidad de codificación más grande o unidad de filtro, según indican los indicadores alf_one_luma_unit_flag, alf_one_cb_unit_flag y alf_one_cr_unit_flag respectivamente en la Figura 15. La estructura de sintaxis también puede incluir una indicación, tal como alf_repeat_row_flag[cIdx], sobre si los parámetros ALF de los bloques de codificación en la fila de bloques de codificación actual son los mismos que los del bloque de codificación superior para el índice de componente de color respectivo, cIdx.

La Figura 16 ilustra una estructura de sintaxis de ejemplo, alf_unit( ), para bloques de codificación ALF según una forma de realización de la presente invención. La sintaxis alf_unit( ) incluye información asociada con información de compartición de parámetros ALF entre bloques de codificación, tales como alf_run_diff y alf_merge_up_flag. El número de veces que parámetros ALF correspondientes a un bloque de codificación se repiten para bloques de codificación subsiguientes en la misma fila está representado por alfRun[cIdx][rx][ry]. El índice de matriz cIdx especifica el componente de color; cIdx tiene un valor igual a 0, 1 o 2 correspondiente a luma, Cb o Cr respectivamente. Los índices rx y ry especifican la ubicación del bloque de codificación subyacente en relación con el bloque de codificación superior izquierdo de la imagen. El elemento de sintaxis, alf_run_diff especifica el alfRun[ ][ ][ ][ ] del bloque de codificación actual si la fila actual es la primera línea, o bien especifica la diferencia entre la serie (run) del bloque de codificación actual y la serie (run) del bloque de codificación superior. Cuando alfRun[ ][ ][ ][ ] es mayor o igual que cero, los elementos de sintaxis de alf_info() se derivan a partir de correspondientes elementos de sintaxis del bloque de codificación izquierdo. La longitud del elemento de sintaxis alf_run_diff es de Ceil(Log2(alf_num_lcu_in_width_minus1 – rx + 2)) bits. Cuando el indicador, alf_merge_up_flag es igual a uno, los elementos de sintaxis de alf_info() se derivan a partir de correspondientes elementos de sintaxis del bloque de codificación superior. Cuando el indicador, alf_merge_up_flag es igual a cero, los elementos de sintaxis de alf_info() no se derivan a partir de correspondientes elementos de sintaxis del bloque de codificación superior. Cuando alf_merge_up_flag no está presente, se infiere que es igual a cero.

La Figura 17 ilustra una estructura de sintaxis de ejemplo para información ALF, alf_info() según una forma de realización de la presente invención. La sintaxis alf_info( ) incluye información asociada al filtro ALF. El diseño de sintaxis de ejemplo admite el uso de un búfer ALF, de modo que puede utilizarse un índice para seleccionar uno de entre múltiples filtros ALF almacenados en el búfer ALF. Por ejemplo, cuando alf_new_filter_set_flag es igual a uno, indica que el bloque de codificación actual utiliza un nuevo conjunto de filtros. De lo contrario, el bloque de codificación actual utiliza el conjunto de filtros almacenados con el índice de búfer igual a alf_stored_filter_set_idx[cIdx] Cuando alf_new_filter_set_flag no está presente, se infiere que es igual a uno. Cuando alf_new_filter_set_flag es igual a uno, se incrementa NumALFFiltersInStoredBuffer[cIdx] en uno, en el que NumALFFiltersInStoredBuffer[cIdx] es el número de filtros en el conjunto de filtros. La sintaxis alf_stored_filter_set_idx[cIdx] especifica el índice de búfer de los filtros almacenados para el componente de color cIdx. La longitud del elemento de sintaxis alf_stored_filter_set_idx[cIdx] es de Floor(Log2(Min(1, NumALFFiltersInStoredBuffer[cIdx] - 1)))) + 1 bits. El elemento de sintaxis alf_no_filters_minus1 se utiliza para derivar el número de conjuntos de filtros para el bloque de codificación actual, en el que alf_no_filters_minus1 tiene

un valor igual a 0 hasta 2. El elemento de sintaxis alf_start_second_filter especifica el índice de modo adaptativo de bloque (BA) de muestras de luma a las que se aplica el segundo filtro. Cuando el elemento de sintaxis alf_filter_pattern_flag[cIdx][ry][rx][rx][i] es igual a uno, se incrementa en uno el índice de filtro para el i-ésimo índice del modo BA. Cuando el elemento de sintaxis alf_filter_pattern_flag[cIdx][ry][rx][rx][i] es igual a cero, el índice de filtro para el i-ésimo índice del modo BA es el mismo que el (i-1)-ésimo índice del modo BA. Cuando el elemento de sintaxis alf_pred_flag[ ][ ][ ][ ][ ]] es igual a uno, los coeficientes de filtro para el bloque de codificación actual se codifican de manera predictiva; de lo contrario, los coeficientes de filtro se codifican directamente. El elemento de sintaxis alf_min_kstart_minus1 + 1 especifica el orden mínimo k del código exponencial de Golomb de orden k-ésimo para los coeficientes de filtro de luma para el filtro de bucle adaptativo. El elemento de sintaxis alf_golomb_index_flag[i] especifica la diferencia en el orden de los códigos de Golomb de orden k-ésimo entre un grupo i-ésimo y un grupo (i+1)-ésimo de los coeficientes de filtro de luma. Hay múltiples grupos de los coeficientes del filtro luma en los que cada grupo puede tener un orden k diferente. Cuando el elemento de sintaxis alf_nb_pred_luma_flag[cIdx][ry][rx][i] es igual a uno, se codifican los coeficientes de filtro del i-ésimo filtro para el bloque de codificación actual de manera predictiva en base a coeficientes de filtro vecinos espacialmente; de lo contrario, no se codifican los coeficientes de filtro usando coeficientes de filtro vecinos espacialmente. El elemento de sintaxis alf_filt_coeff[cIdx][ry][rx][i] especifica el coeficiente de filtro j-ésimo del filtro i-ésimo utilizado en el proceso de filtrado de bucle adaptativo para el bloque de codificación actual.

Se puede implementar una forma de realización de sistemas de codificación de vídeo que incorporan un procesamiento SAO y/o ALF mejorado de acuerdo con la presente invención, tal como se ha descrito anteriormente, en diversos hardware, códigos de software o una combinación de ambos. Por ejemplo, una forma de realización de la presente invención puede ser un circuito integrado en un chip de compresión de vídeo o códigos de programa integrados en software de compresión de vídeo para realizar el procesamiento descrito en este documento. Una forma de realización de la presente invención también puede ser códigos de programa para su ejecución en un Procesador de Señal Digital (DSP) para realizar el procesamiento descrito en este documento. La invención también puede involucrar un número de funciones a realizar por un procesador informático, un procesador de señal digital, un microprocesador, o una matriz de puertas programables de campo (FPGA). Estos procesadores pueden ser configurados para realizar tareas particulares según la invención, ejecutando código software legible informáticamente o código firmware que define los procedimientos particulares representados por la invención. Los códigos de software o de firmware pueden ser desarrollados en diferentes lenguajes de programación y en diferentes formatos o estilos. El código de software también puede ser compilado para diferentes plataformas de destino. Aun así, diferentes formatos de código, estilos y lenguajes de códigos de software y otros medios de configurar código para realizar las tareas de acuerdo con la invención no se apartarán del espíritu y alcance de la invención.

Los ejemplos descritos deben considerarse en todos los aspectos sólo como ilustrativos y no restrictivos. El alcance de la invención es por tanto, indicado por las reivindicaciones adjuntas en lugar de por la descripción anterior. Todos los cambios que entren en el significado y rango de equivalencia de las reivindicaciones deben ser incluidos en su alcance.

En aras de la compleción, se describe la presente invención en base a las siguientes cláusulas:

1. Un procedimiento para decodificación de vídeo con un procesamiento en bucle de un vídeo reconstruido, comprendiendo el procedimiento:
recuperar datos de vídeo reconstruidos a partir de un flujo de bits de vídeo;
recibir un indicador procedente del flujo de bits de vídeo;
según el indicador, recibir información asociada con parámetros de filtro en bucle, ya sea procedente de una carga útil de datos en el flujo de bits de vídeo para su compartición por dos o más bloques de codificación o procedente de datos de bloques de codificación individuales en el flujo de bits de vídeo; y
aplicar el procesamiento en bucle a bloques de codificación del vídeo reconstruido.

2. El procedimiento de la cláusula 1, en el que el procesamiento en bucle corresponde a un Filtro de Bucle Adaptativo (ALF) o a un Desplazamiento Adaptativo de muestra (SAO).

3. El procedimiento de la cláusula 1, en el que el bloque de codificación corresponde a una unidad de codificación (CU), múltiples CU, una unidad de codificación más grande (LCU) o múltiples LCU.

4. El procedimiento de la cláusula 1, en el que la carga útil de datos en el flujo de bits de vídeo se encuentra en un nivel de imagen, un conjunto de parámetros de adaptación (APS) o una cabecera de segmento.

5. El procedimiento de la cláusula 1, en el que el indicador es un indicador de intercalación, en el que el indicador de intercalación se refiere a si la información asociada con los parámetros de filtro en bucle está incorporada en la

carga útil de datos en el flujo de bits de vídeo o intercalada con datos de bloques de codificación individuales en el flujo de bits de vídeo.

6. El procedimiento de la cláusula 5, en el que existen dos indicadores de intercalación en al menos dos cargas útiles de datos correspondientes a al menos dos miembros diferentes seleccionados de entre un grupo compuesto por un nivel de imagen, un conjunto de parámetros de adaptación (APS) o una cabecera de segmento.

7. El procedimiento de la cláusula 6, en el que los dos indicadores de intercalación tienen el mismo valor.

8. El procedimiento de la cláusula 6, en el que los dos indicadores de intercalación existen en el APS y en la cabecera de segmento; en el que los dos indicadores de intercalación tienen un valor que indica que la información asociada con los parámetros del filtro en bucle está incorporada en el APS; y cada segmento incluye un identificador APS que referencia al APS para permitir que los bloques de codificación en el segmento utilicen la información asociada con los parámetros de filtro en bucle incorporados en el APS.

9. El procedimiento de la cláusula 1, en el que el indicador se encuentra en un nivel de secuencia y en el que la información asociada con los parámetros del filtro en bucle está incorpora en el conjunto de parámetros de adaptación (APS), o una cabecera de segmento según el indicador.

10. El procedimiento de la cláusula 9, en el que el procesamiento en bucle corresponde a un Filtro de bucle adaptativo (ALF); la cabecera de segmento comprende información asociada con parámetros ALF si el ALF está habilitado y el indicador indica que la información asociada con los parámetros ALF está en la cabecera de segmento; y la cabecera de segmento comprende un parámetro de control de unidad de codificación si el ALF está habilitado y el indicador indica que la información asociada con los parámetros ALF está en el APS.

11. Un procedimiento para codificación de vídeo con un procesamiento en bucle de un vídeo reconstruido, comprendiendo el procedimiento:

recibir datos de vídeo reconstruidos;

determinar parámetros de filtro en bucle asociados con el procesamiento en bucle, en el que el procesamiento en bucle es aplicado a bloques de codificación del vídeo reconstruido; e

incorporar información asociada con los parámetros del filtro en bucle, ya sea en una carga útil de datos en un flujo de bits de vídeo para su compartición por dos o más bloques de codificación, o intercalada con datos de bloques de codificación individuales en el flujo de bits de vídeo según un indicador.

12. El procedimiento de la cláusula 11, en el que el procesamiento en bucle corresponde a un Filtro de Bucle Adaptativo (ALF) o a un Desplazamiento Adaptativo de muestra (SAO).

13. El procedimiento de la cláusula 11, en el que el bloque de codificación corresponde a una unidad de codificación (CU), múltiples CU, una unidad de codificación más grande (LCU) o múltiples LCU.

14. El procedimiento de la cláusula 11, en el que la carga útil de datos en el flujo de bits de vídeo se encuentra en un nivel de imagen, un conjunto de parámetros de adaptación (APS) o una cabecera de segmento.

15. El procedimiento de la cláusula 11, en el que existen dos indicadores en al menos dos cargas útiles de datos correspondientes a al menos dos miembros diferentes seleccionados de entre un grupo compuesto por un nivel de imagen, un conjunto de parámetros de adaptación (APS) o una cabecera de segmento.

16. El procedimiento de la cláusula 15, en el que los dos indicadores tienen el mismo valor.

17. El procedimiento de la cláusula 15, en el que los dos indicadores existen en el APS y en la cabecera de segmento; en el que los dos indicadores tienen un valor que indica que la información asociada con los parámetros del filtro en bucle es incorporada en el APS; y cada segmento incluye un identificador de APS que referencia al APS para permitir que los bloques de codificación en el segmento utilicen la información asociada con los parámetros del filtro en bucle incorporados en el APS.

18. El procedimiento de la cláusula 11, en el que el indicador se encuentra en un nivel de secuencia y en el que la información asociada con los parámetros del filtro en bucle es incorporada en el conjunto de parámetros de adaptación (APS), o en un cabecera de segmento según el indicador.

19. El procedimiento de la cláusula 18, en el que el procesamiento en bucle corresponde a un Filtro de bucle adaptativo (ALF); la cabecera de segmento incluye información asociada con parámetros ALF si el ALF está habilitado y el indicador indica que la información asociada con los parámetros ALF está en la cabecera de

segmento; y la cabecera de segmento incluye un parámetro de control de unidad de codificación si el ALF está habilitado y el indicador indica que la información asociada con los parámetros ALF está en el APS.

20. Un aparato para decodificación de vídeo con un procesamiento en bucle de un vídeo reconstruido, comprendiendo el aparato:
medios para recuperar datos de vídeo reconstruidos a partir de un flujo de bits de vídeo;
medios para recibir un indicador procedente del flujo de bits de vídeo;
medios para recibir información asociada con parámetros de filtro en bucle ya sea procedente de una carga útil de datos en el flujo de bits de vídeo para su compartición por dos o más bloques de codificación o procedente de datos de bloques de codificación individuales en el flujo de bits de vídeo según el indicador; y
medios para aplicar el procesamiento en bucle a bloques de codificación del vídeo reconstruido.

21. Un aparato para codificación de vídeo con un procesamiento en bucle de un vídeo reconstruido, comprendiendo el aparato:
medios para recibir datos de vídeo reconstruidos;
medios para determinar parámetros de filtro en bucle asociados con el procesamiento en bucle, en el que el procesamiento en bucle es aplicado a bloques de codificación del vídeo reconstruido; y
medios para incorporar información asociada con los parámetros del filtro en bucle, ya sea en una carga útil de datos en un flujo de bits de vídeo para su compartición por dos o más bloques de codificación, o intercalada con datos de bloques de codificación individuales en el flujo de bits de vídeo según un indicador.

22. Un procedimiento para decodificación de vídeo con un procesamiento de desplazamiento adaptativo de muestra (SAO) de un vídeo reconstruido, comprendiendo el procedimiento:
recuperar datos de vídeo reconstruidos a partir de un flujo de bits de vídeo;
recibir información asociada con parámetros SAO procedente de datos de bloques de codificación individuales en el flujo de vídeo; y
aplicar el procesamiento SAO a los bloques de codificación del vídeo reconstruido;
en el que la información asociada con los parámetros SAO está intercalada con datos de bloques de codificación individuales en el flujo de vídeo.

23. El procedimiento de la cláusula 22, en el que el bloque de codificación corresponde a una unidad de codificación (CU), múltiples CU, una unidad de codificación más grande (LCU) o múltiples LCU.

24. Un procedimiento para codificación de vídeo con un procesamiento de desplazamiento adaptativo de muestra (SAO) de un vídeo reconstruido, comprendiendo el procedimiento:
recibir datos de vídeo reconstruidos;
determinar parámetros SAO asociados con el procesamiento SAO, en el que el procesamiento SAO es aplicado a bloques de codificación del vídeo reconstruido; e
incorporar información asociada con los parámetros SAO intercalándola con datos de bloques de codificación individuales en un flujo de vídeo.

25. El procedimiento de la cláusula 24, en el que el bloque de codificación corresponde a una unidad de codificación (CU), múltiples CU, una unidad de codificación más grande (LCU) o múltiples LCU.

La presente invención también puede ser descrita por las siguientes cláusulas:

1. Un procedimiento para decodificación de vídeo con un procesamiento de desplazamiento adaptativo de muestra (SAO) de un vídeo reconstruido, comprendiendo el procedimiento:
recuperar datos de vídeo reconstruidos a partir de un flujo de bits de vídeo;
recibir información asociada con parámetros SAO procedente de datos de bloques de codificación individuales en el flujo de vídeo; y
aplicar el procesamiento SAO a bloques de codificación del vídeo reconstruido;
en el que la información asociada con los parámetros SAO está intercalada con datos de bloques de codificación individuales en el flujo de vídeo.

2. El procedimiento de la cláusula 1, en el que el bloque de codificación corresponde a una unidad de codificación (CU), múltiples CU, una unidad de codificación más grande (LCU) o múltiples LCU.

3. Un procedimiento para codificación de vídeo con un procesamiento de desplazamiento adaptativo de muestra (SAO) de un vídeo reconstruido, comprendiendo el procedimiento:
recibir datos de vídeo reconstruidos;

determinar parámetros SAO asociados con el procesamiento SAO, en el que el procesamiento SAO es aplicado a bloques de codificación del vídeo reconstruido; e

incorporar información asociada con los parámetros SAO intercalándola con datos de bloques de codificación individuales en un flujo de vídeo.

4. El procedimiento de la cláusula 3, en el que el bloque de codificación corresponde a una unidad de codificación (CU), múltiples CU, una unidad de codificación más grande (LCU) o múltiples LCU.

5. Un procedimiento para decodificación de vídeo con un procesamiento de desplazamiento adaptativo de muestra (SAO) de un vídeo reconstruido, comprendiendo el procedimiento:

recuperar datos de vídeo reconstruidos a partir de un flujo de bits de vídeo;

recibir información asociada con parámetros SAO procedente de datos de bloques de codificación individuales en el flujo de vídeo; y

aplicar el procesamiento SAO a bloques de codificación de los datos de vídeo reconstruidos, en el que la información asociada con parámetros SAO es compartida entre al menos dos componentes de color diferentes.

6. El procedimiento de la cláusula 5, en el que los parámetros SAO incluyen un indicador de fusión, un indicador de fusión izquierda, un indicador de fusión superior, un tipo de SAO, un offset SAO, o una combinación de los mismos.

7. El procedimiento de la cláusula 6, en el que el tipo de SAO comprende uno o una combinación de habilitación de procesamiento SAO, deshabilitación de procesamiento SAO, desplazamiento de banda y desplazamiento de borde.

8. El procedimiento de la cláusula 5, en el que los componentes de color comprenden un componente de color de luma, un componente de color de croma Cb y un componente de color de croma Cr.

9. El procedimiento de la cláusula 5, que comprende además derivar un parámetro SAO de un componente de color a partir de un parámetro SAO de otro componente de color para el procesamiento SAO.

10. Un procedimiento para codificación de vídeo con un procesamiento de desplazamiento adaptativo de muestra (SAO) de un vídeo reconstruido, comprendiendo el procedimiento:

recibir datos de vídeo reconstruidos;

determinar parámetros SAO para el procesamiento SAO, en el que el procesamiento SAO es aplicado a bloques de codificación de los datos de vídeo reconstruidos; e

incorporar información asociada con los parámetros SAO en un flujo de bits de vídeo, en el que la información es compartida entre al menos dos componentes de color diferentes.

11. El procedimiento de la cláusula 10, en el que los parámetros SAO comprenden un indicador de fusión, un indicador de fusión izquierda, un indicador de fusión superior, un tipo de SAO, un desplazamiento SAO o una combinación de los mismos.

12. El procedimiento de la cláusula 11, en el que el tipo de SAO comprende uno o una combinación de habilitación de procesamiento SAO, deshabilitación de procesamiento SAO, desplazamiento de banda y desplazamiento de borde.

13. El procedimiento de la cláusula 10, en el que los componentes de color comprenden un componente de color de luma, un componente de color de croma Cb, un componente de color de croma Cr.

14. El procedimiento de la cláusula 10, en el que determinar los parámetros SAO incluye además derivar un parámetro SAO de un componente de color a partir de un parámetro SAO de otro componente de color.

REIVINDICACIONES

1. Un procedimiento para codificación de vídeo con un procesamiento de desplazamiento adaptativo de muestra, SAO, de un vídeo reconstruido, comprendiendo el procedimiento:
recibir datos de vídeo reconstruidos;
aplicar el procesamiento SAO a unas regiones (A, B, C, D) de los datos de vídeo reconstruidos según un indicador SAO; y
codificar o decodificar según los datos de vídeo reconstruidos procesados por el SAO;
caracterizado porque
el indicador SAO es un indicador SAO de croma que indica si el procesamiento SAO se aplica al menos a un componente de croma de los datos de vídeo reconstruidos.

2. El procedimiento de la reivindicación 1, en el que la información SAO para un primer componente de croma es derivada a partir de información SAO para un segundo componente de croma.

3. El procedimiento de la reivindicación 2, en el que la información SAO comprende un tipo de SAO y una decisión ON/OFF o activación/desactivación.

4. Un aparato para codificación de vídeo con un procesamiento de desplazamiento adaptativo de muestra, SAO, de un vídeo reconstruido, comprendiendo el aparato uno o más circuitos electrónicos configurados para:
recibir datos de vídeo reconstruidos;
aplicar el procesamiento SAO a unas regiones (A, B, C, D) de los datos de vídeo reconstruidos según un indicador SAO; y
codificar o decodificar según los datos de vídeo reconstruidos procesados por el SAO;
caracterizado porque
el indicador SAO es un indicador SAO de croma que indica si el procesamiento SAO se aplica al menos a un componente de croma de los datos de vídeo reconstruidos.

5. Flujo de datos que tiene codificado un indicador SAO en el mismo,
caracterizado porque
el indicador SAO es un indicador SAO de croma que indica si el procesamiento SAO se aplica al menos a un componente de croma de los datos de vídeo reconstruidos.

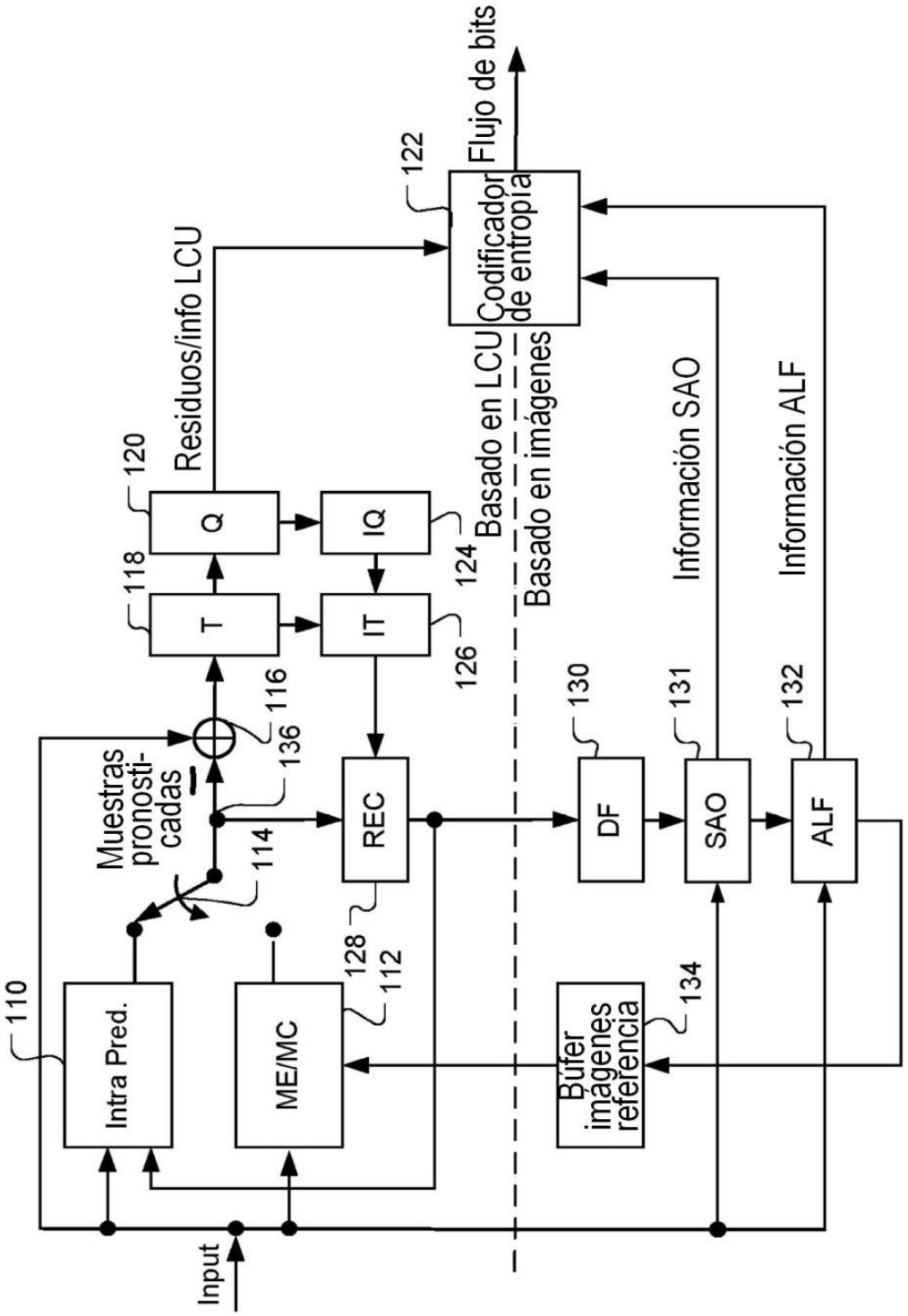
Flujo de bits
122
Codificador de entropía
Residuos/info LCU
120
Q
IQ
124
Basado en LCU
Basado en imágenes
118
T
IT
126
116
136
Muestras pronosti-cadas
114
REC
130
DF
131
Información SAO
132
Información ALF
SAO
ALF
128
112
110
Intra Pred.
ME/MC
Búfer imágenes referencia
134
Input

Fig. 1A

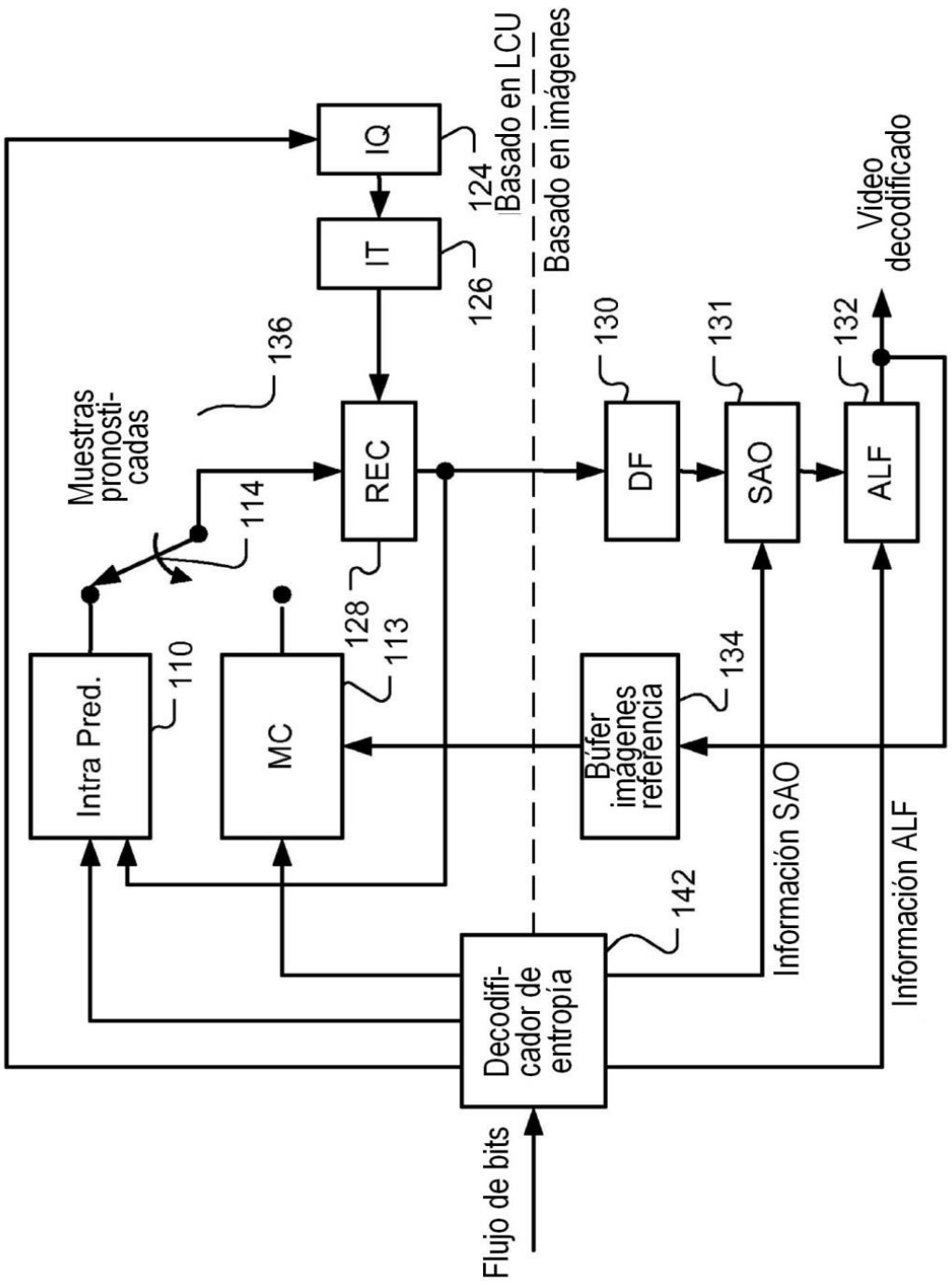
IQ
124
Basado en LCU
Basado en imágenes
IT
126
Video decodificado
136
Muestras pronosti-cadas
REC
130
DF
131
SAO
132
ALF
114
128
113
Intra Pred.
110
MC
Búfer imágenes referencia
134
142
Información SAO
Información ALF
Decodifi-cador de entropía
Flujo de bits

Fig. 1B

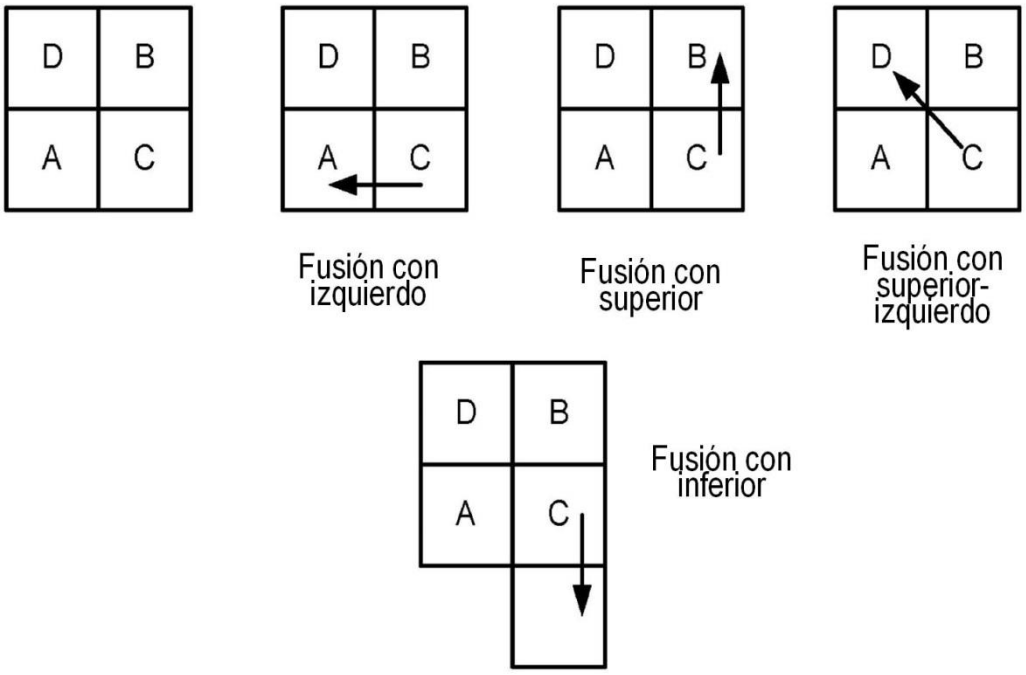
D
B
A
C
Fusión con izquierdo
Fusión con superior
Fusión con superior-izquierdo
Fusión con inferior

Fig. 2

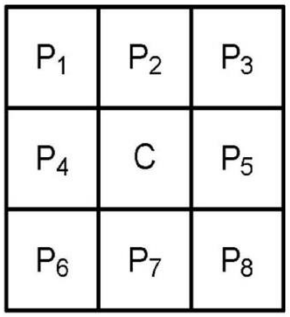
P1
P2
P3
P4
C
P5
P6
P7
P8

Fig. 3

seq_parameter_set_rbsp( ) {	**Descriptor**
…	.
loop_filter_across_slice_flag	u(1)
sample_adaptive_offset_enabled_flag	u(1)
…	
}	

Fig. 4

aps_rbsp( ) {	**Descriptor**
aps_id	ue(v)
…	
aps_sao_interleaving_flag	u(1)
if(!aps_sao_interleaving_flag) {	
aps_sample_adaptive_offset_flag	u(1)
if(aps_sample_adaptive_offset_flag)	
aps_sao_param( )	
}	
…	
}	

Fig. 5

aps_sao_param( ) {	**Descriptor**
sao_cb_enable_flag	u(1)
sao_cr_enable_flag	u(1)
sao_num_lcu_in_width_minus1	ue(v)
sao_one_luma_unit_flag	u(1)
if (sao_one_luma_unit_flag)	
sao_offset_vlc(0, 0, 0)	
if (sao_cb_enable_flag){	
sao_one_cb_unit_flag	u(1)
if (sao_one_cb_unit_flag)	
sao_offset_vlc(0, 0, 1)	
}	
if (sao_cr_enable_flag){	
sao_one_cr_unit_flag	u(1)
if (sao_one_cr_unit_flag)	
sao_offset_vlc(0, 0, 2)	
}	
for (ry = 0; ry < sao_num_lcu_in_height_minus1+1; ry++) {	
for (rx = 0; rx < sao_num_lcu_in_width_minus1+1; rx++) {	
if (aps_sample_adaptive_offset_flag && ! sao_one_luma_unit_flag) {	
if (ry > 0 && rx == 0)	
sao_repeat_row_flag[0]	u(1)
sao_unit_vlc(rx, ry, 0)	
}	
if (sao_cb_enable_flag && ! sao_one_cb_unit_flag) {	
if (ry > 0 && rx == 0)	
sao_repeat_row_flag[1]	u(1)
sao_unit_vlc(rx, ry, 1)	
}	
if (sao_cr_enable_flag && !sao_one_cr_unit_flag) {	
if (ry > 0 && rx == 0)	
sao_repeat_row_flag[2]	u(1)
sao_unit_vlc(rx, ry, 2)	
}	
}	
}	
}	

Fig. 6

sao_unit_vlc(rx, ry, cIdx){	**Descriptor**
if (!sao_repeat_row_flag[cIdx]) {	
if (rx == 0 || run[cIdx][rx][ry] < 0) {	
if (ry == 0)	
saoRun[cIdx][rx][ry] = **sao_run_diff**	u(v)
else	
saoRun[cIdx][rx][ry] = **sao_run_diff** + saoRun[cIdx][rx][ry − 1]	se(v)
}	
saoRun[cIdx][rx + 1][ry] = saoRun[cIdx][rx][ry] – 1	
if (rx == 0 || saoRun[cIdx][rx][ry] < 0) {	
if (ry > 0)	
sao_merge_up_flag	u(1)
if (!sao_merge_up_flag)	
sao_offset_vlc(rx, ry, cIdx)	
}	
} else {	
saoRun[cIdx][ry][rx] = saoRun[cIdx][ry − 1][rx]	
}	
}	

Fig. 7

sao_offset_vlc(rx, ry, cIdx) {	**Descriptor**
sao_type_idx[cIdx][rx][ry]	ue(v)
if(sao_type_idx[cIdx][rx][ry] = =5) {	
sao_band_position[cIdx][rx][ry]	u(5)
for(i = 0; i < 4; i++)	
sao_offset[cIdx][rx][ry][i]	se(v)
}	
else if(sao_type_idx[cIdx][rx][ry] != 0)	
for(i = 0; i < 4; i++)	
sao_offset[cIdx][rx][ry][i]	ue(v)
}	

Fig. 8

slice_header( ) {	**Descriptor**
…	
if(scaling_list_enable_flag || deblocking_filter_in_APS_enabled_flag || sample_adaptive_offset_enabled_flag || adaptive_loop_filter_enabled_flag) {	
if(sample_adaptive_offset_enabled_flag) {	
slice_sao_interleaving_flag	u(1)
slice_sample_adaptive_offset_flag	u(1)
if(slice_sao_interleaving_flag) {	
if(slice_sample_adaptive_offset_flag) {	
sao_cb_enable_flag	u(1)
sao_cr_enable_flag	u(1)
}	
}	
}	
aps_id	ue(v)
}	
…	
}	

Fig. 9

slice_data( ) {	Descriptor
…	
do {	
xCtb = horizontal location of largest coding unit	
yCtb = vertical location largest coding unit	
…	
CtbAddrInSlice = first largest coding unit location in current slice	
If above largest coding unit is available set AddrUp larger than 0, otherwise, set AddrUp = -1	
if(slice_sao_interleaving_flag) {	
if(slice_sample_adaptive_offset_flag)	
sao_unit_cabac(xCtb, yCtb, 0)	
if(sao_cb_enable_flag)	
sao_unit_cabac(xCtb, yCtb, 1)	
if(sao_cr_enable_flag)	
sao_unit_cabac(xCtb, yCtb, 2)	
}	
…	
} while(next largest coding unit data is exist)	
}	

Fig. 10

sao_unit_cabac(rx, ry, cIdx){	Descriptor
if (rx > 0 && CtbAddrInSlice != 0)	
sao_merge_left_flag	ae(v)
if (!sao_merge_left_flag) {	
if (ry > 0 && (AddrUp > 0 || loop_filter_across_slice_flag))	
sao_merge_up_flag	ae(v)
if (!sao_merge_up_flag)	
sao_offset_cabac(rx, ry, cIdx)	
}	
}	

Fig. 11

sao_offset_cabac(rx, ry, cIdx) {	**Descriptor**
sao_type_idx[cIdx][rx][ry]	ae(v)
if(sao_type_idx[cIdx][rx][ry] = =5)	
sao_band_position[cIdx][rx][ry]	ae(v)
if(sao_type_idx[cIdx][rx][ry] != 0)	
for(i = 0; i < 4; i++)	
sao_offset[cIdx][rx][ry][i]	ae(v)
}	

Fig. 12

seq_parameter_set_rbsp( ) {	**Descriptor**
profile_idc	u(8)
reserved_zero_8bits /* equal to 0 */	u(8)
level_idc	u(8)
seq_parameter_set_id	ue(v)
.....	
adaptive_loop_filter_enabled_flag	u(1)
if(adaptive_loop_filter_enabled_flag)	
alf_coef_in_slice_flag	u(1)
.....	
}	

Fig. 13

slice_header( ) {	**Descriptor**
first_slice_in_pic_flag	u(1)
if(first_slice_in_pic_flag = = 0)	
slice_address	u(v)
slice_type	ue(v)
…..	
if(adaptive_loop_filter_enabled_flag) {	
slice_adaptive_loop_filter_flag	u(1)
if(slice_adaptive_loop_filter_flag && alf_coef_in_slice_flag)	
alf_param( )	
if(slice_adaptive_loop_filter_flag && !alf_coef_in_slice_flag)	
alf_cu_control_param( )	
}	
……..	
}	

Fig. 14

alf_param() {	**Descriptor**
alf_cb_enable_flag	u(1)
alf_cr_enable_flag	u(1)
alf_one_luma_unit_per_slice_flag	
if(alf_cb_enable_flag)	
alf_one_cb_unit_per_slice_flag	u(1)
if(alf_cr_enable_flag)	
alf_one_cr_unit_per_slice_flag	
if(!alf_coef_in_slice_flag) {	
alf_num_lcu_in_width_minus1	ue(v)
alf_num_lcu_in_height_minus1	ue(v)
} else {	
alf_num_lcu_in_slice_minus1	u(v)
}	
endCtbrY = (numCtb – 1 + firstCtbAddr) / numCtbInWidth	
endCtbrX = (numCtb – 1 + firstCtbAddr) % numCtbInWidth	
for (i = 0; i < numCtb; i++) {	
rx = (i + firstCtbAddr) % numCtbInWidth	
ry = (i + firstCtbAddr) / numCtbInWidth	
endrX = (ry = = endCtbrY) ? (endCtbrX) : (numCtbInWidth – 1)	
if((rx = = 0) && (i − numCtbInWidth >= 0) && (alf_one_luma_unit_per_slice_flag = =0))	
alf_repeat_row _flag[0]	u(1)
alf_unit (rx, ry, 0, i, endrX, alf_one_luma_unit_per_slice_flag)	
if(alf_cb_enable_flag) {	
if((rx = = 0) && (i − numCtbInWidth >= 0) && (alf_one_cb_unit_per_slice_flag = =0))	
alf_repeat_row _flag[1]	u(1)
alf_unit (rx, ry, 1, i, endrX, alf_one_cb_unit_per_slice_flag)	
}	
if(alf_cr_enable_flag) {	
if((rx = = 0) && (i − numCtbInWidth>= 0) && (alf_one_cr_unit_per_slice_flag = =0))	
alf_repeat_row _flag[2]	u(1)
alf_unit (rx, ry, 2, i, endrX, alf_one_cr_unit_per_slice_flag)	
}	
}	
}	

Fig. 15

alf_unit(rx, ry, cIdx, lcuIdx, endrX, oneUnitFlag) {	**Descriptor**
if(oneUnitFlag) {	
if (lcuIdx = = 0) {	
alf_lcu_enable_flag[cIdx][ry][rx]	u(1)
if(alf_lcu_enable_flag[cIdx][ry][rx])	
alf_info(rx, ry, cIdx)	
}	
} else {	
if(!alf_repeat_row _flag[cIdx]) {	
if(rx = = 0 || alfRun[cIdx][ry][rx] < 0) {	
if(lcuIdx –numCtbInWidth < 0)	
alfRun[cIdx][ry][rx] = 0 + **alf_run_diff**	u(v)
else	
alfRun[cIdx][ry][rx] = alfRun[cIdx][ry – 1][rx]+ **alf_run_diff**	s(v)
}	
alfRun[cIdx][ry][rx+ 1] = alfRun[cIdx][ry][rx] – 1	
if (rx == 0 || alfRun [cIdx] [rx][ry] < 0) {	
if (ry > 0 && (lcuIdx − numCtbInWidth > = 0 | | alfAcrossSlice))	
alf_merge_up_flag	u(1)
if(!alf_merge_up_flag) {	
alf_lcu_enable_flag[cIdx][ry][rx]	u(1)
if(alf_lcu_enable_flag[cIdx][ry][rx])	
alf_info (rx, ry, cIdx)	
}	
}	
} else	
alfRun[cIdx][ry][rx] = alfRun[cIdx][ry – 1][rx]	
}	
}	

Fig. 16

alf_info (rx, ry, cIdx) {	**Descriptor**
if(NumALFFiltersInStoredBuffer[cIdx] > 0)	
alf_new_filter_set_flag	u(1)
if(alf_new_filter_set_flag = = 0 && NumALFFiltersInStoredBuffer[cIdx] >	
alf_stored_filter_set_idx[cIdx]	u(v)
} else {	
if(cIdx = = 0) {	
alf_no_filters_minus1	ue(v)
if(alf_no_filters_minus1 = = 1) {	
alf_start_second_filter	ue(v)
} else if(alf_no_filters_minus1 > 1) {	
for (i = 1; i < 15; i++)	
alf_filter_pattern_flag[cIdx][ry][rx][i]	u(1)
}	
if(alf_no_filters_minus1 > 0)	
alf_pred_flag[cIdx][ry][rx]	u(1)
for (i = 0; i < AlfNumFilters; i++)	
alf_nb_pred_luma_flag[cIdx][ry][rx][i]	u(1)
if(AlfNumFilters > 1) {	
alf_min_kstart_minus1	ue(v)
for (i = 1; i < 4; i++)	
alf_golomb_index_flag[i]	u(1)
}	
for (i = 0; i < AlfNumFilters; i++) {	
for (j = 0; j < AlfCodedLength; j++)	
alf_filt_coeff[cIdx][ry][rx][i][j]	ge(v)
}	
} else {	
for (j = 0; j < AlfCodedLength; j++)	
alf_filt_coeff[cIdx][ry][rx][0][j]	se(v)
}	
}	
}	

Fig. 17