(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 791 420**

(51) Int. CI.:

G06F 7/533 (2006.01)
G06F 7/483 (2006.01)
G06F 7/499 (2006.01)
G06F 17/14 (2006.01)
H04N 19/60 (2014.01)
H04N 19/42 (2014.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **28.08.2008 PCT/US2008/074607**

(87) Fecha y número de publicación internacional: **12.03.2009 WO09032740**

(96) Fecha de presentación y número de la solicitud europea: **28.08.2008 E 08829843 (5)**

(97) Fecha y número de publicación de la concesión europea: **11.03.2020 EP 2195750**

(54) Título: **Cálculo rápido de productos por fracciones diádicas con errores de redondeo de signo simétrico**

(30) Prioridad:

28.08.2007 US 968381 P
16.06.2008 US 139957

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
04.11.2020

(73) Titular/es:

QUALCOMM INCORPORATED (100.0%)
Attn: International IP Administration 5775 Morehouse Drive
San Diego, CA 92121, US

(72) Inventor/es:

REZNIK, YURIY

(74) Agente/Representante:

FORTEA LAGUNA, Juan José

DESCRIPCIÓN

Cálculo rápido de productos por fracciones diádicas con errores de redondeo de signo simétrico

ANTECEDENTES

Campo

[0001] La materia en el presente documento se refiere en general al procesamiento, y más en particular a las técnicas de aproximación usadas en el procesamiento de hardware y programas informáticos.

Antecedentes

[0002] Los desplazamientos aritméticos se pueden usar para realizar la multiplicación o división de enteros con signo por potencias de dos. El desplazamiento a la izquierda en n bits en un número binario con signo o sin signo tiene el efecto de multiplicarlo por 2n. El desplazamiento a la derecha en n bits en un número binario con signo del complemento a dos tiene el efecto de dividirlo por 2n, pero siempre se redondea hacia abajo (es decir, hacia el infinito negativo). Debido a que los desplazamientos a la derecha no son operaciones lineales, los desplazamientos aritméticos a la derecha pueden añadir errores de redondeo y producir resultados que pueden no ser iguales al resultado de la multiplicación seguida por el desplazamiento a la derecha.

[0003] En algunas implementaciones, el algoritmo de signo simétrico se puede usar en una arquitectura de transformada IDCT u otro filtro digital.

[0004] Un ejemplo del uso de desplazamientos aritméticos es en implementaciones de punto fijo de algunos algoritmos de procesamiento de señales, tales como FFT, DCT, MLT, MDCT, etc. Dichos algoritmos de procesamiento de señales típicamente aproximan factores irracionales (algebraicos o trascendentales) en definiciones matemáticas de estos algoritmos usando fracciones racionales diádicas. Esto permite que las multiplicaciones por estas fracciones irracionales se realicen usando sumas y desplazamientos de enteros, en lugar de operaciones más complejas.

[0005] Reznik, Y. A., Hinds, A. T., Zhang, C., Yu, L., y Ni, Z., Efficient fixed-point approximations of the 8×8 inverse discrete cosine transform. Proc SPIE. 6696, 2007, divulgan metodologías de diseño de punto fijo y varias implementaciones resultantes de la Transformada de coseno discreta inversa (IDCT) contribuyeron al trabajo de MPEG en la definición de la nueva norma IDCT de punto fijo 8x8 - ISO/IEC 23002-2. También se describe el algoritmo especificado en el Borrador del Comité Final (FCD) de esta norma.

[0006] El documento WO 2007/047478 A2 divulga técnicas para realizar cálculos para el procesamiento de señales y datos. Para el procesamiento sin multiplicación, se genera una serie de valores intermedios basados en un valor de entrada para que los datos que se van a procesar. Se genera al menos un valor intermedio en la serie basado en al menos otro valor intermedio en la serie. Se proporciona un valor intermedio en la serie como valor de salida para una multiplicación del valor de entrada con un valor constante. El valor constante puede ser una constante entera, una constante racional o una constante irracional. Una constante irracional se puede aproximar con una constante diádica racional que tiene un numerador entero y un denominador que es una potencia de dos. El procesamiento sin multiplicación se puede usar para diversas transformadas (por ejemplo, DCT e IDCT), filtros y otros tipos de procesamiento de señales y datos.

[0007] Hinds, A. T., Methodology for the Design of Linear Algorithms for Signal Processing Applications. Ph.D. Dissertation. Universidad de Colorado en Boulder, Boulder, CO, EE. UU., 2007 , divulga una metodología para diseñar implementaciones de punto fijo de algoritmos lineales para aplicaciones de procesamiento de señales digitales. Los algoritmos seleccionados son algoritmos de cálculo intensivo cuyas definiciones requieren el uso de constantes de punto flotante, tales como IDCT y DCT directo.

[0008] Rijavec, N. y Hinds, AT, Multicriterial Optimization Approach to Eliminating Multiplications, 2006 IEEE Workshop on Multimedia Signal Processing, 3 al 6 de octubre de 2006, Victoria, BC, págs. 368-371 , divulgan que los algoritmos de compresión de imagen, tal como DCT, se pueden describir como la aplicación de un operador lineal a un vector de muestras. Supuestamente, se han desarrollado un número de algoritmos para minimizar el número de operaciones necesarias, en particular se multiplica. Se describe un procedimiento para encontrar aproximaciones de las constantes de multiplicación restantes en estos algoritmos, de modo que el error de aproximación se minimiza supuestamente para cada sustitución de multiplicaciones por desplazamientos y sumas. El documento US 2002/152251 A1 divulga un aparato y un procedimiento para el filtrado digital que incluye un procedimiento para implementar un filtro digital que tiene coeficientes de filtro, cada uno de ellos expresable como una palabra de código de dígitos con signo canónico; donde el procedimiento incluye formar una subexpresión común virtual que sea pertinente para un primer coeficiente de filtro, formando una segunda subexpresión para un segundo coeficiente de filtro en términos de la subexpresión común virtual para que los sumadores se compartan

con la subexpresión común virtual en una línea de derivación de los segundos coeficientes de filtro; y el filtro digital resultante recibe muestras digitales de señales de entrada, desplaza las muestras digitales recibidas por valores de desplazamiento de bits de coeficientes de filtro que se definen en relación con la subexpresión común virtual, añade muestras digitales desplazadas para conducir una línea de derivación común, añade muestras digitales desplazadas a la salida de la línea de derivación común para conducir una línea de derivación correspondiente a un coeficiente de filtro, y retrasa un componente de señal de salida correspondiente a una línea de derivación.

BREVE EXPLICACIÓN

[0009] La invención se define mediante las reivindicaciones independientes. Las reivindicaciones dependientes definen modos de realización ventajosos.

[0010] Un producto de un valor entero y un valor irracional se puede determinar mediante un algoritmo de signo simétrico. Un proceso puede determinar posibles algoritmos que minimizan las métricas, tal como la asimetría media, el error medio, la varianza de error y la magnitud de error. Dada la variable entera x y las constantes diádicas racionales que aproximan la fracción irracional, se pueden producir una serie de valores intermedios que son de signo simétrico. Dada una secuencia de operaciones de suma, resta y desplazamiento a la derecha, el algoritmo de signo simétrico puede aproximar el producto del entero y el valor irracional. Se pueden eliminar otras operaciones, tales como sumas o restas de 0s o desplazamientos en 0 bits para simplificar el procesamiento.

[0011] Esta breve explicación se proporciona para introducir una selección de conceptos de forma simplificada que se describen adicionalmente a continuación en la descripción detallada. Esta breve explicación no pretende identificar rasgos característicos clave o rasgos característicos esenciales de la materia reivindicada, ni se pretende usar para limitar el alcance de la materia reivindicada.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

[0012]

La FIG. 1 es un gráfico de resultados de diversos algoritmos computacionales.

La FIG. 2 es un diagrama de flujo de un proceso de ejemplo para determinar un algoritmo de signo simétrico para determinar un producto.

La FIG. 3 es una arquitectura ejemplar que implementa un algoritmo IDCT de punto fijo.

La FIG. 4 ilustra un diagrama de bloques de un sistema de codificación ejemplar.

La FIG. 5 ilustra un diagrama de bloques de un sistema de descodificación ejemplar.

DESCRIPCIÓN DETALLADA

[0013] Las transformadas de coseno discretas (DCT) y las transformadas de coseno discretas inversas (IDCT) realizan operaciones de multiplicación con constantes irracionales (es decir, los cosenos). En el diseño de implementaciones de las DCT/IDCT, las aproximaciones de los productos de computación de estas constantes irracionales se pueden realizar usando aritmética de punto fijo. Una técnica para convertir valores de punto flotante a punto fijo se basa en las aproximaciones de factores irracionales α_i por fracciones diádicas:

$$\alpha_i \approx a_i/2^k \qquad (1)$$

donde tanto $_i$ como k son enteros. La multiplicación de x por el factor α_i proporciona una implementación de una aproximación en aritmética de enteros, como sigue:

$$x\alpha_i \approx (x * a_i) >> k \qquad (2)$$

donde >> indica la operación de desplazamiento a la derecha a nivel de bits.

[0014] El número de bits de precisión, k, puede afectar la complejidad de las aproximaciones racionales diádicas. En implementaciones de programa informático, el parámetro de precisión k puede estar restringido por el ancho de los registros (por ejemplo, 16 o 32) y la consecuencia de no satisfacer dicha restricción de diseño puede dar como resultado la extensión del tiempo de ejecución de la transformada. En los diseños de hardware, el parámetro de precisión k afecta el número de puertas necesarias para implementar sumadores y multiplicadores. Por lo tanto, un objetivo en los diseños de punto fijo es minimizar el número total de bits k, mientras se mantiene una precisión suficiente de aproximaciones.

[0015] Sin ninguna restricción específica a los valores de α_i, y suponiendo que para cualquier k dado, los valores correspondientes de los nominadores a_i se pueden elegir de modo que:

$$\left|\alpha_i - a_i/2^k\right| = 2^{-k}\left|2^k\alpha_i - a_i\right| = 2^{-k}\min_{z\in Z}\left|2^k\alpha_i - z\right|$$

y el error absoluto de aproximaciones en (1) debe ser inversamente proporcional a 2^k:

$$|\alpha_i - a_i/2^k| \leq 2^{-k-1}$$

Es decir, cada bit adicional de precisión (es decir, incrementando k), debería reducir el error a la mitad.

[0016] En algunas implementaciones, la tasa de error se puede mejorar si los valores α_1, αn que se van a aproximar se pueden escalar mediante algún parámetro adicional ξ. Si $\alpha_1,\ldots,\alpha_n$ son un conjunto de n números irracionales (n ≥ 2), luego, existen infinitas n+2-tuplas $a_1,\ldots,a_n$, k, ξ, con $a_1,\ldots,a_n$ e Z, k e N y ξ ∈ Q, de modo que:

$$máx\ \{|\xi\alpha_1 - a_1/2^k|,\ldots,|\xi\alpha_n - a_n/2^k|\} < \frac{n}{n+1}\xi^{-1/n}2^{-k(1+1/n)}$$

[0017] En otras palabras, si el algoritmo se puede alterar de modo que todos sus factores irracionales $\alpha_1,\ldots,\alpha_n$ se puedan preescalar mediante algún parámetro ξ, luego deben existir aproximaciones que tengan un error absoluto que disminuya tan rápido como 2-k(1+1/n). Por ejemplo, cuando n=2, pueden existir una efectividad aproximadamente un 50% mayor en el uso de bits. Para conjuntos grandes de factores $\alpha_1,\ldots,\alpha_n$, sin embargo, esta ganancia puede ser más pequeña.

[0018] Las aproximaciones diádicas que se muestran en las relaciones (1, 2) anteriores, reducen el problema del cálculo de productos de computación por constantes irracionales a multiplicaciones por enteros. La multiplicación de un entero por un factor irracional $\frac{1}{\sqrt{2}}$, usando su aproximación diádica de 5 bits 23/32, ilustra el proceso de aproximación de constantes irracionales. Observando el patrón de bits binarios de 23 = 10111 y sustituyendo cada "1" con una operación de suma, el producto de un entero multiplicado por 23 se puede determinar como sigue:

$$x*\ 23 = (x<<4) + (x<<2) + (x<<1) + x$$

[0019] Esta aproximación requiere 3 operaciones de suma y 3 operaciones de desplazamiento. Al observar además que los últimos 3 dígitos forman una serie de "1"s, se puede usar lo siguiente:

$$x*\ 23 = (x<<4) + (x<<3) - x,$$

lo que reduce la complejidad a solo 2 operaciones de desplazamiento y 2 de suma.

[0020] Las secuencias de operaciones "+" asociadas con dígitos aislados "1", o "+" y "-" asociados con comienzos y finales de secuencias "1 ...1" se denominan comúnmente una descomposición de "dígitos con signo canónico" (CSD). CSD es una implementación bien conocida en el diseño de circuitos sin multiplicador. Sin embargo, las descomposiciones CSD no siempre producen resultados con el menor número de operaciones. Por ejemplo, considerando una aproximación de 8 bits del mismo factor

$1/\sqrt{2} \approx 181/256 = 10110101$ y su descomposición CSD:

$$x * 181 = (x<<7) + (x<<5) + (x<<4) + (x<<2) + x$$

que usa 4 operaciones de suma y 4 de desplazamiento. Reorganizando los cálculos y reutilizando los resultados intermedios, se puede construir un algoritmo más eficaz:

```
x2= x +(x<<2);          //          101
x3= x2+(x<<4);          //        10100
x4= x3+(x2 <<5);        //     10110101      = x * 181
```

[0021] De acuerdo con una implementación, el cálculo de productos por fracciones diádicas se puede derivar permitiendo el uso de desplazamientos a la derecha como operaciones elementales. Por ejemplo, considerando

un factor $1/\sqrt{2} \approx 23/32 = 10111,$ y usando las operaciones de desplazamiento a la derecha y de suma de acuerdo con su descomposición CSD, se obtiene lo siguiente:

$$x* 23/32 \sim (x>>1) + (x>>2) - (x>>5). \quad (3)$$

o señalando además que 1/2 + 1/4 = 1 - 1/4:

$$x* 23/32 \sim x- (x>>2) - (x>>5). \quad (4)$$

[0022] Otra forma de calcular el producto por el mismo factor es:

$$x* 23/32 \sim x- ((x+(x>>4)) >>2) + ((-x) >>6) . \quad (5)$$

[0023] La FIG. 1 ilustra las gráficas de valores producidos por algoritmos (3, 4 y 5) frente a la multiplicación de un entero y la fracción irracional 23/32. Cada algoritmo (3, 4 y 5) calcula valores que aproximan productos multiplicados por la fracción irracional 23/32; sin embargo, los errores en cada una de estas aproximaciones son diferentes. Por ejemplo, el algoritmo (4) produce todos los errores positivos, con una magnitud máxima de 55/32. El algoritmo (3) tiene errores más equilibrados, con la magnitud de las oscilaciones dentro de ±65/64. Finalmente, el algoritmo (5) produce errores perfectamente simétricos en signos con oscilaciones en ±7/8. Por tanto, un algoritmo de signo simétrico producirá resultados equilibrados que minimizan los errores.

[0024] La propiedad de simetría de signos de un algoritmo

$$A_{a_i,b}(x) \rightsquigarrow xa_i/2^b$$

significa que para cualquier (x e Z):

$$A_{a_i,b}(-x) = -A_{a_i,b}(x)$$

y también implica que para cualquier N, y suponiendo que $A_{ai,b}(0) = 0$:

$$\sum_{x=-N}^{N} \left[A_{a_i,b}(x) - x\frac{a_i}{b}\right] = 0$$

es decir, un error de media cero en cualquier intervalo simétrico.

[0025] Esta propiedad se puede usar en el diseño de algoritmos de procesamiento de señales, ya que minimiza la probabilidad de que se acumulen errores de redondeo introducidos por aproximaciones de punto fijo. A continuación se describe la base de los algoritmos de signo simétrico basados en desplazamiento a la derecha para calcular productos por fracciones diádicas, así como los límites superiores para su complejidad.

[0026] Dado un conjunto de fracciones diádicas $a_{1/2}{}^b,...,a_m/2^b$, se puede definir un algoritmo:

$$A_{ai,...,am,b}(x) \rightsquigarrow (xa_1/2^b ,..., xa_m/2^b)$$

como la siguiente secuencia de etapas:

$$x_1,x_2,...,x_t,$$

donde $x_1 := x$, y donde los valores posteriores x_k (k =2,...,t) se producen usando una de las siguientes operaciones elementales:

$$x_k := \begin{cases} x_i >> s_k; & 1 \leqslant i < k,\ s_k \geqslant 1; \\ -x_i; & 1 \leqslant i < k; \\ x_i + x_j; & 1 \leqslant i, j < k; \\ x_i - x_j; & 1 \leqslant i, j < k,\ i \neq j. \end{cases}$$

[0027] El algoritmo termina cuando existen índices $j_1,...,j_m \leq t$, de modo que:

$$x_{j1} \sim x * a_1/2^b, \ldots, x_{jm} \sim x * a_m/2^b$$

[0028] Por tanto, algunas implementaciones examinan algoritmos que minimizan una o más de las siguientes métricas:

asimetría media:

$$\chi_{A_{a_i,b}} = \frac{1}{2^b}\sum_{x=1}^{2^b}\left|A_{a_i,b}(x) + A_{a_i,b}(-x)\right|$$

error medio:

$$\mu_{A_{a_i,b}} = \frac{1}{2^{b+1}}\sum_{x=-2^b}^{2^b}\left[A_{a_i,b}(x) - x\frac{a_i}{2^b}\right]$$

varianza de error:

$$\sigma^2_{A_{a_i,b}} = \frac{1}{2^{b+1}-1}\sum_{x=-2^b}^{2^b}\left[A_{a_i,b}(x) - \mu_{A_{a_i,b}}\right]^2$$

magnitud de error:

$$\delta^{máx}_{A_{a_i,b}} = \underset{x=-2^b \ldots 2^b}{máx}\left|A_{a_i,b}(x) - \mu_{A_{a_i,b}}\right|$$

[0029] Al calcular productos por múltiples constantes $\frac{a_1}{2^b}, \ldots, \frac{a_m}{2^b}$, los valores del peor caso de las métricas anteriores (calculadas para cada una de las constantes $\frac{a_1}{2^b}, \ldots, \frac{a_m}{2^b}$) se puede usar para evaluar la eficacia del algoritmo.

[0030] La FIG. 2 muestra etapas en un proceso 100 para calcular un producto. En 102, se recibe un valor entero y, en 104, se determinan las constantes diádicas racionales representativas del valor irracional que se va a multiplicar por el entero. En 106, se pueden determinar los valores intermedios. Por ejemplo, dada la variable de entero *x*y un conjunto de constantes diádicas racionales $\frac{a_1}{2^b}, \ldots, \frac{a_m}{2^b}$, se puede determinar una serie de valores intermedios como sigue:

$$w_0, w_1, w_2, \ldots, w_t$$

donde: $w_0 = 0$, $w_1 = x$, y para todos los valores $k \geq 2$ se obtienen w_k como sigue:

$$w_k = \pm w_i \pm \left(w_j >> s_k\right)(i, j < k),$$

donde los signos ± implican una operación de más o de menos que se debe realizar con ambos términos, y >> implica el desplazamiento a la derecha de la variable z_j en s_k bits.

[0031] En 108, se determinan los puntos en esta serie correspondientes a productos. Es decir, el resultado de esta etapa son los índices $l_1, \ldots, l_m \leq t$, de modo que:

$$w_{l_1} \approx x\frac{a_1}{2^b}, \ldots, w_{l_m} \approx x\frac{a_m}{2^b}$$

[0032] En 110, los valores de salida resultantes $w_{l_1} \approx x\frac{a_1}{2^b}, \ldots, w_{l_m} \approx x\frac{a_m}{2^b}$ se analizan frente a ciertas métricas de precisión. Por ejemplo, estos valores se pueden analizar para determinar si minimizan una de la media, asimetría, varianza, magnitud.

[0033] En algunas implementaciones, el proceso 100 puede eliminar sumas o restas de 0s, o desplazamientos en 0 bits. En algunas implementaciones, la secuencia de valores intermedios se puede elegir de modo que el costo computacional (o de implementación) total de esta operación entera sea mínimo.

[0034] Por tanto, dado un conjunto de métricas, pueden existir algoritmos que tengan una complejidad que se pueda caracterizar por un número total de sumas, un número total de desplazamientos, etc. Como tal, existe un algoritmo que tiene un menor número de sumas, un menor número de desplazamientos, un menor número de sumas y desplazamientos, y un menor número de sumas entre los algoritmos que logran el menor número de sumas y desplazamientos, etc.

[0035] La FIG. 3 ilustra una arquitectura IDCT 8x8 de punto fijo ejemplar 120. La arquitectura puede implementar algoritmos que son valores de signo simétrico de x. En muchas implementaciones, dichos algoritmos pueden ser los menos complejos para un conjunto de factores dado. Como se indica anteriormente, el diseño de los IDCT puede ser simétrico o dar como resultado errores de redondeo bien equilibrados. En algunas implementaciones, se pueden analizar estimaciones de métricas tales como la media, la varianza y la magnitud (valores máximos) de errores producidos por los algoritmos. Al evaluar la complejidad de los algoritmos, se puede tener en cuenta los números de operaciones, así como la ruta de ejecución más larga y el número máximo de registros intermedios necesarios para los cálculos.

[0036] En algunas implementaciones, la arquitectura usada en el diseño de la arquitectura IDCT de punto fijo 120 propuesta se puede caracterizar teniendo rasgos característicos separables y escalados. Una etapa de ajuste a escala 122 puede incluir una única matriz de 8x8 que se calcula previamente factorizando los factores de escala 1D para la transformada de fila con los factores de escala 1D para la transformada de columna. La etapa de ajuste a escala 122 también se puede usar para preasignar P bits de precisión a cada uno de los coeficientes DCT de entrada, proporcionando de este modo una "mantisa" de punto fijo para su uso en el resto de la transformada.

[0037] En una implementación, la base para el diseño de transformada 1D escalada puede ser una variante de la factorización bien conocida de C. Loeffler, A. Ligtenberg y G.S. Moschytz con 3 rotaciones planas y 2 factores independientes $\gamma = \sqrt{2}$. Para proporcionar aproximaciones racionales eficaces de las constantes α, β, δ, ε, η y θ dentro de la factorización LLM, se pueden usar dos factores flotantes ξ y ζ, y aplicarlos a dos subgrupos de estas constantes como sigue:

$$\xi : \alpha' = \xi\alpha,\ \beta' = \xi\beta;$$

$$\zeta : \delta' = \zeta\delta,\ \varepsilon = \zeta\varepsilon,\ \eta' = \zeta\eta,\ \theta' = \zeta\theta;$$

[0038] Estas multiplicaciones se pueden invertir por ξ y ζ en la etapa de ajuste a escala 122 multiplicando cada coeficiente DCT de entrada con el respectivo recíproco de ξ y ζ. Es decir, se puede calcular un vector de factores de escala para su uso en la etapa de ajuste a escala 122 antes de la primera en la cascada de transformadas ID (por ejemplo, etapas 126 y 128):

$$\sigma = (1,\ 1/\zeta,\ 1/\xi,\ \gamma/\zeta,\ 1,\ \gamma/\zeta,\ 1/\xi,\ 1/\zeta)^T$$

[0039] Estos factores se pueden fusionar posteriormente en una matriz de ajuste a escala que se calcula previamente como sigue:

$$\Sigma = \sigma\sigma^T 2^S = \begin{pmatrix} A & B & C & D & A & D & C & B \\ B & E & F & G & B & G & F & E \\ C & F & H & I & C & I & H & F \\ D & G & I & J & D & J & I & G \\ A & B & C & D & A & D & C & B \\ D & G & I & J & D & J & I & G \\ C & F & H & I & C & I & H & F \\ B & E & F & G & B & G & F & E \end{pmatrix}$$

donde A - J indica valores únicos en este producto:

$$A = 2^S,\ B = \frac{2^S}{\zeta},\ C = \frac{2^S}{\xi},\ D = \frac{\gamma 2^S}{\zeta},\ E = \frac{2^S}{\zeta^2},\ F = \frac{2^S}{\xi\zeta},\ G = \frac{\gamma 2^S}{\zeta^2},\ H = \frac{2^S}{\xi^2},\ I = \frac{\gamma 2^S}{\xi\zeta},\ J = \frac{\gamma^2 2^S}{\zeta^2},$$

y S indica el número de bits de precisión de punto fijo asignados para el ajuste a escala.

[0040] Este parámetro S se puede elegir de modo que sea mayor o igual que el número de bits P para la mantisa de cada coeficiente de entrada. Esto permite el ajuste a escala de los coeficientes F_{vu}, que se implementará como sigue:

$$F'_{vu} = (F_{vu} * S_{vu}) >> (S - P)$$

donde $S_{vu} \approx \Sigma vu$ indica aproximaciones enteras de valores en la matriz de factores de escala.

[0041] Al final de la última etapa de transformada en la serie de transformadas ID (etapas 126 y 128), los bits de mantisa de punto fijo P (más 3 bits adicionales acumulados durante las ejecuciones de cada una de las etapas de ID‡) simplemente se desplazan fuera de las salidas de transformada mediante operaciones de desplazamiento a la derecha 130, como sigue:

$$f_{yx} = f'_{yx} >> (P + 3)$$

[0042] Para garantizar un redondeo apropiado del valor calculado, se puede añadir un sesgo de 2P+2 a los valores f'_{yx} antes de los desplazamientos usando una etapa de sesgo DC 124. Este sesgo de redondeo se implementa perturbando el coeficiente DC antes de ejecutar la primera transformada 1D:

$$F''_{00} = F'_{00} + 2^{P+2}$$

[0043] En algunas implementaciones, los algoritmos equilibrados (es decir, de signo simétrico) como se analiza anteriormente se pueden usar en la norma ISO/IEC 23002-2 IDCT. Esta norma define el proceso para el cálculo de productos mediante las siguientes constantes:

$$y \sim y * 113/128,$$

$$z \sim y * 719/4096$$

y se logra como sigue:

```
x2=(x>>3)-(x>>7);
x3=x2-(x>>11);
y=x2+(x3>>1);
z=x-x2;
```

[0044] La FIG. 4 muestra un diagrama de bloques de un sistema de codificación 400, que puede incluir transformadas que implementan las fracciones diádicas que tienen errores de redondeo de signo simétrico, como se describe anteriormente. Un dispositivo de captura/una memoria 410 puede recibir una señal fuente, realizar la conversión a formato digital, y proporciona datos de entrada/sin procesar. El dispositivo de captura 410 puede ser una cámara de vídeo, un digitalizador o algún otro dispositivo. Un procesador 420 procesa los datos sin procesar y genera datos comprimidos. Dentro del procesador 420, los datos sin procesar pueden ser transformados por una unidad DCT 422, escaneados por una unidad de exploración en zigzag 424, cuantificados por un cuantificador 426, codificados por un codificador por entropía 428, y empaquetados por un empaquetador 430. La unidad DCT 422 puede realizar DCT 2D en los datos sin procesar de acuerdo con las técnicas descritas en el presente documento y puede soportar interfaces tanto completas como escaladas. Cada una de las unidades 422 a 430 se puede implementar con hardware, firmware y/o programa informático. Por ejemplo, la unidad DCT 422 se puede implementar con hardware dedicado, un conjunto de instrucciones para una unidad de lógica aritmética (ALU), etc.

[0045] Una unidad de almacenamiento 440 puede almacenar los datos comprimidos del procesador 420. Un transmisor 442 puede transmitir los datos comprimidos. Un controlador/procesador 450 controla el funcionamiento de diversas unidades en el sistema de codificación 400. Una memoria 452 almacena datos y códigos de programa para el sistema de codificación 400. Uno o más buses 460 interconectan diversas unidades en el sistema de codificación 400.

[0046] La FIG. 5 muestra un diagrama de bloques de un sistema de descodificación 500, que puede incluir transformadas que implementan las fracciones diádicas que tienen errores de redondeo de signo simétrico, como se describe anteriormente. Un receptor 510 puede recibir datos comprimidos de un sistema de codificación, y una unidad de almacenamiento 512 puede almacenar los datos comprimidos recibidos. Un procesador 520 procesa los datos comprimidos y genera datos de salida. Dentro del procesador 520, los datos comprimidos pueden ser desempaquetados por un desempaquetador 522, descodificados por un descodificador por entropía 524, cuantificados de manera inversa por un cuantificador inverso 526, colocados en el orden apropiado por una unidad de exploración en zigzag inversa 528, y transformados por una unidad IDCT 530. La unidad IDCT 530 puede realizar IDCT 2D en los coeficientes de transformada completos o escalados de acuerdo con las técnicas descritas en el presente documento y puede soportar interfaces tanto completas como escaladas. Cada una de las unidades

522 a 530 se puede implementar con hardware, firmware y/o programa informático. Por ejemplo, la unidad IDCT 530 se puede implementar con hardware dedicado, un conjunto de instrucciones para una ALU, etc.

[0047] Una unidad de visualización 540 muestra imágenes reconstruidas y vídeo del procesador 520. Un controlador/procesador 550 controla el funcionamiento de diversas unidades en el sistema de descodificación 500. Una memoria 552 almacena datos y códigos de programa para el sistema de descodificación 500. Uno o más buses 560 interconectan diversas unidades en el sistema de descodificación 500.

[0048] Los procesadores 420 y 520 se pueden implementar cada uno con uno o más circuitos integrados de aplicación específica (ASIC), procesadores digitales de señales (DSP) y/o algún otro tipo de procesadores. De forma alternativa, los procesadores 420 y 520 se pueden reemplazar cada uno con una o más memorias de acceso aleatorio (RAM), memoria de solo lectura (ROM), ROM programables eléctricas (EPROM), ROM programables borrables eléctricamente (EEPROM), discos magnéticos, discos ópticos y/u otros tipos de memorias volátiles y no volátiles conocidos en la técnica.

[0049] Los modos de realización descritos en el presente documento se pueden implementar por hardware, programa informático, firmware, middleware, microcódigo o cualquier combinación de los mismos. Cuando los sistemas y/o procedimientos se implementan en programa informático, firmware, middleware o microcódigo, código de programa o segmentos de código, se pueden almacenar en un medio legible por máquina, tal como un componente de almacenamiento. Un segmento de código puede representar un procedimiento, una función, un subprograma, un programa, una rutina, una subrutina, un módulo, un paquete de programa informático, una clase o cualquier combinación de instrucciones, estructuras de datos o sentencias de programa. Un segmento de código se puede acoplar a otro segmento de código o a un circuito de hardware pasando y/o recibiendo información, datos, argumentos, parámetros o contenido de memoria. La información, los argumentos, los parámetros, los datos, etc., se pueden pasar, reenviar o transmitir usando cualquier medio adecuado, incluyendo la compartición de memoria, el paso de mensajes, el paso de testigos, la transmisión de red, etc.

[0050] En una implementación en programa informático, las técnicas descritas en el presente documento se pueden implementar con módulos (por ejemplo, procedimientos, funciones, etc.) que realizan las funciones descritas en el presente documento. Los códigos de programa informático se pueden almacenar en unidades de memoria y ejecutar por procesadores. La unidad de memoria se puede implementar dentro del procesador o de manera externa al procesador, en cuyo caso se puede acoplar de manera comunicativa al procesador mediante diversos medios, como se conoce en la técnica.

[0051] Las etapas de un procedimiento o algoritmo descrito en relación con los modos de realización divulgados en el presente documento se pueden llevar a cabo directamente en hardware, en un módulo de programa informático ejecutado por un procesador o en una combinación de ambos. Un módulo de programa informático puede residir en memoria de acceso aleatorio (RAM), memoria flash, memoria de solo lectura (ROM), memoria de solo lectura programable borrable (EPROM), memoria de solo lectura programable borrable eléctricamente (EEPROM), registros, un disco duro, un disco extraíble, un CD-ROM o en cualquier otra forma de medio de almacenamiento conocida en la técnica. Un medio de almacenamiento de ejemplo está acoplado al procesador de modo que el procesador pueda leer información de, y escribir información en, el medio de almacenamiento. De forma alternativa, el medio de almacenamiento puede estar integrado en el procesador. El procesador y el medio de almacenamiento pueden residir en un circuito de usuario de aplicación específica (ASIC). El ASIC puede residir en un terminal de usuario. De forma alternativa, el procesador y el medio de almacenamiento pueden residir como componentes discretos en un terminal de usuario.

[0052] Cabe señalar que los procedimientos descritos en el presente documento se pueden implementar en una variedad de hardware, procesadores y sistemas conocidos por un experto en la técnica. Por ejemplo, una máquina que se usa en una implementación puede tener una pantalla de visualización para mostrar contenido e información, un procesador para controlar el funcionamiento del cliente y una memoria para almacenar datos y programas relacionados con el funcionamiento de la máquina. En algunas implementaciones, la máquina es un teléfono celular. En algunas implementaciones, la máquina es un ordenador de mano o un teléfono que tiene capacidades de comunicación. En otra implementación, la máquina es un ordenador personal que tiene capacidades de comunicación.

[0053] Los diversos lógicas, bloques lógicos, módulos y circuitos ilustrativos descritos en relación con las implementaciones divulgadas en el presente documento se pueden implementar o realizar con un procesador de propósito general, un DSP, un ASIC, una matriz de puertas programable in situ (FPGA) u otro dispositivo de lógica programable, lógica de transistores o puertas discretas, componentes de hardware discretos, o cualquier combinación de los mismos diseñada para realizar las funciones descritas en el presente documento. Un procesador de uso general puede ser un microprocesador pero, de forma alternativa, el procesador puede ser cualquier procesador, controlador, microcontrolador o máquina de estados convencional. Un procesador también puede implementarse como una combinación de dispositivos informáticos, por ejemplo, una combinación de un DSP y un microprocesador, una pluralidad de microprocesadores, uno o más microprocesadores junto con un núcleo de DSP o cualquier otra configuración de este tipo.

REIVINDICACIONES

1. Un procedimiento para mantener una exactitud de aproximaciones para una señal fuente proporcionada en un formato digital para su uso en algoritmos de procesamiento de señales determinando un algoritmo de procesamiento de señales que minimiza una métrica, que comprende las etapas de:

recibir un valor entero x;

determinar un conjunto de fracciones diádicas $a_i/2^b$,..., $a_m/2^b$ que aproximan factores irracionales α_1, ..., α_m;

determinar una secuencia de valores intermedios w_1, ..., w_t para calcular una aproximación de productos $x\alpha_1$, ..., $x\alpha_m$ mediante:

establecer w_1 igual al valor entero de entrada x; y

determinar w_2, ..., w_t de acuerdo con

(a) al menos uno de w,..., w_{t-1}, y

(b) una de una operación positiva, una operación negativa y una operación de desplazamiento a la derecha;

determinar índices l_1, ..., $l_m \leq t$ de modo que:

$$w_{l_1} \approx xa_i/2^b, \ldots, w_{l_m} \approx xa_m/2^b;$$

y

determinar la secuencia de valores intermedios y los índices para minimizar la métrica.

2. El procedimiento de la reivindicación 1, en el que la métrica comprende una o más de una métrica de asimetría media, una métrica de error medio, una métrica de varianza de error, y una métrica de magnitud de error.

3. El procedimiento de la reivindicación 2, que comprende además la etapa de:

evaluar una eficiencia del algoritmo de procesamiento de señales en base al resultado del peor caso de acuerdo con la una o más de la métrica de asimetría media, la métrica de error medio, la métrica de varianza de error y la métrica de magnitud de error.

4. El procedimiento de la reivindicación 1, en el que la etapa de determinar la secuencia de valores intermedios y los índices comprende la etapa de determinar la secuencia de valores intermedios que tienen un número menor de sumas.

5. El procedimiento de la reivindicación 1, en el que la etapa de determinar la secuencia de valores intermedios y los índices comprende la etapa de determinar la secuencia de valores intermedios que tienen un número menor de desplazamientos a la derecha.

6. El procedimiento de la reivindicación 1, en el que la etapa de determinar la secuencia de valores intermedios y los índices comprende la etapa de determinar la secuencia de valores intermedios que tienen el menor número de sumas y desplazamientos a la derecha.

7. El procedimiento de la reivindicación 6, en el que la etapa de determinar la secuencia de valores intermedios y los índices comprende la etapa de determinar la secuencia de valores intermedios que tienen un menor número de sumas entre la secuencia de valores intermedios que tienen el menor número de sumas y desplazamientos a la derecha.

8. El procedimiento de la reivindicación 1, en el que la etapa de determinar $w2$, ..., w_t, comprende la etapa de definir un miembro w_k de los valores intermedios como teniendo uno de los valores $w_i >> s_k$, $-w_i$, $w_i + w_j$, o $w_i - w_j$, donde s_k es un número de bits que se van a desplazar a la derecha w_i, i es menor que k, y j es menor que k.

9. El procedimiento de la reivindicación 1, en el que la etapa de determinar la secuencia de valores intermedios y los índices comprende la etapa de determinar una secuencia de signo simétrico como minimizando una métrica de asimetría media.

10. Un medio legible por ordenador que comprende instrucciones almacenados en el mismo que, cuando se ejecutan por un procesador, provocan que el procesador realice el procedimiento de acuerdo con una cualquiera de las reivindicaciones 1 a 9.

11. Aparato configurado para mantener una exactitud de aproximaciones para una señal fuente proporcionada en un formato digital para su uso en algoritmos de procesamiento de señales mediante la determinación de un algoritmo de procesamiento de señales que minimiza una métrica, que comprende:

medios para recibir un valor entero x;

medios para determinar un conjunto de fracciones diádicas $a_i/2^b$, ..., $a_m/2^b$ que aproximan factores irracionales α_1, ..., α_m;

medios para determinar una secuencia de valores intermedios w_1, ..., w_t para calcular una aproximación de los productos $x\alpha_1$, ..., $x\alpha_m$ por:

establecer w_1 igual al valor entero de entrada x; y

determinar w_2, ..., w_t de acuerdo con

(a) al menos uno de w,..., w_{t-1}, y

(b) una de una operación positiva, una operación negativa y una operación de desplazamiento a la derecha;

medios para determinar los índices l_1, ..., $l_m \leq t$ de modo que:

$$w_{l_1} \approx xa_i/2^b, \ldots, w_{l_m} \approx xa_m/2^b;$$

y

medios para determinar la secuencia de valores intermedios y los índices para minimizar la métrica.

12. El aparato de la reivindicación 11, en el que la métrica comprende una o más de una métrica de asimetría media, una métrica de error medio, una métrica de varianza de error y una métrica de magnitud de error.

13. El aparato de la reivindicación 12, que comprende además:

medios para evaluar la eficiencia del algoritmo de procesamiento de señales en base al resultado del peor caso de acuerdo con la una o más de la métrica de asimetría media, la métrica de error medio, la métrica de varianza de error y la métrica de magnitud de error.

14. El aparato de la reivindicación 11, en el que los medios para determinar la secuencia de valores intermedios y los índices comprenden medios para determinar la secuencia de valores intermedios que tienen un número menor de sumas.

15. El aparato de la reivindicación 11, en el que los medios para determinar la secuencia de valores intermedios y los índices comprenden medios para determinar la secuencia de valores intermedios que tienen un número menor de desplazamientos a la derecha.

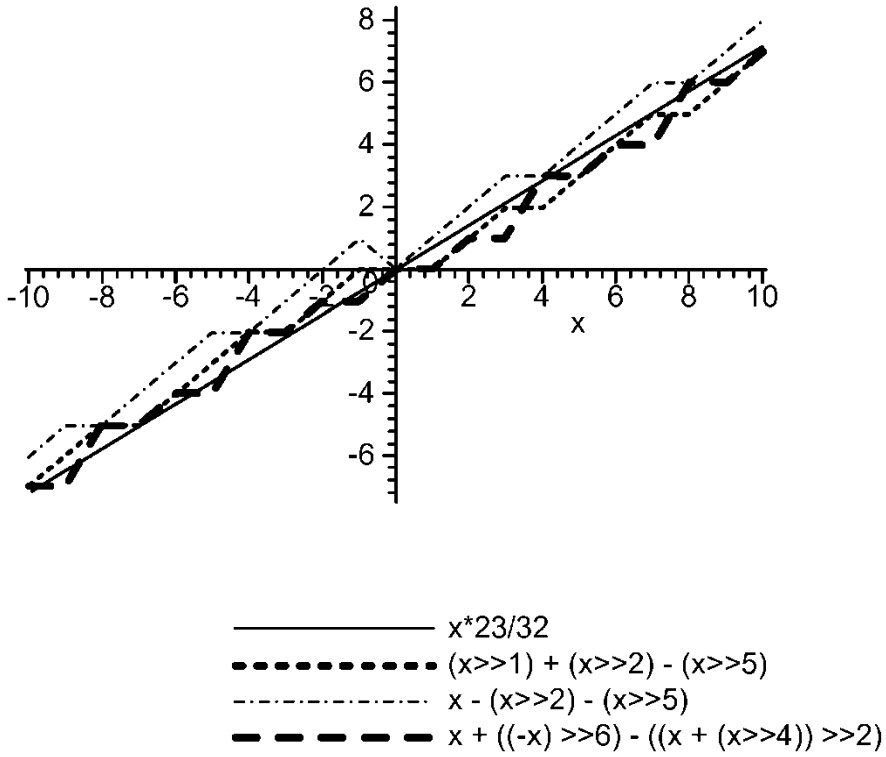

FIG. 1

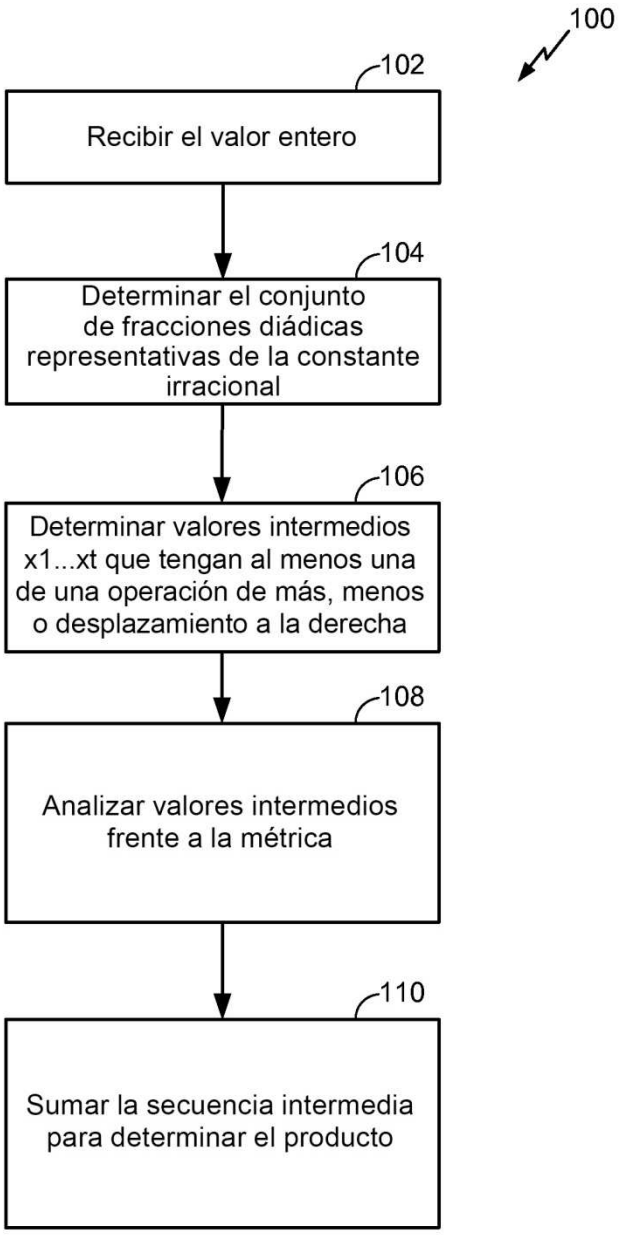
100
102
Recibir el valor entero
104
Determinar el conjunto de fracciones diádicas representativas de la constante irracional
106
Determinar valores intermedios x1...xt que tengan al menos una de una operación de más, menos o desplazamiento a la derecha
108
Analizar valores intermedios frente a la métrica
110
Sumar la secuencia intermedia para determinar el producto

FIG. 2

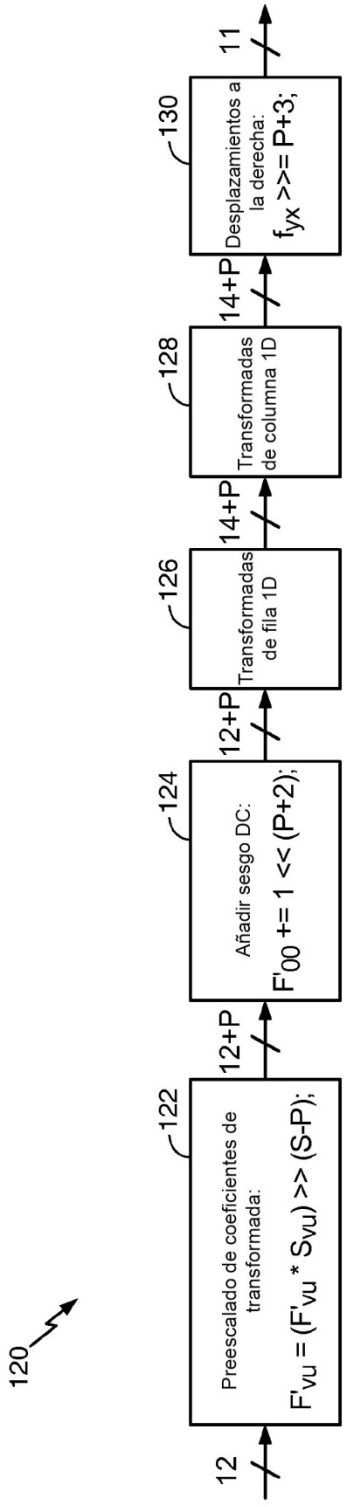
120
12
122
Preescalado de coeficientes de transformada:
F'vu = (F'vu * Svu) >> (S-P);
12+P
124
Añadir sesgo DC:
F'00 += 1 << (P+2);
12+P
126
Transformadas de fila 1D
14+P
128
Transformadas de columna 1D
14+P
130
Desplazamientos a la derecha:
fyx >>= P+3;
11

FIG. 3

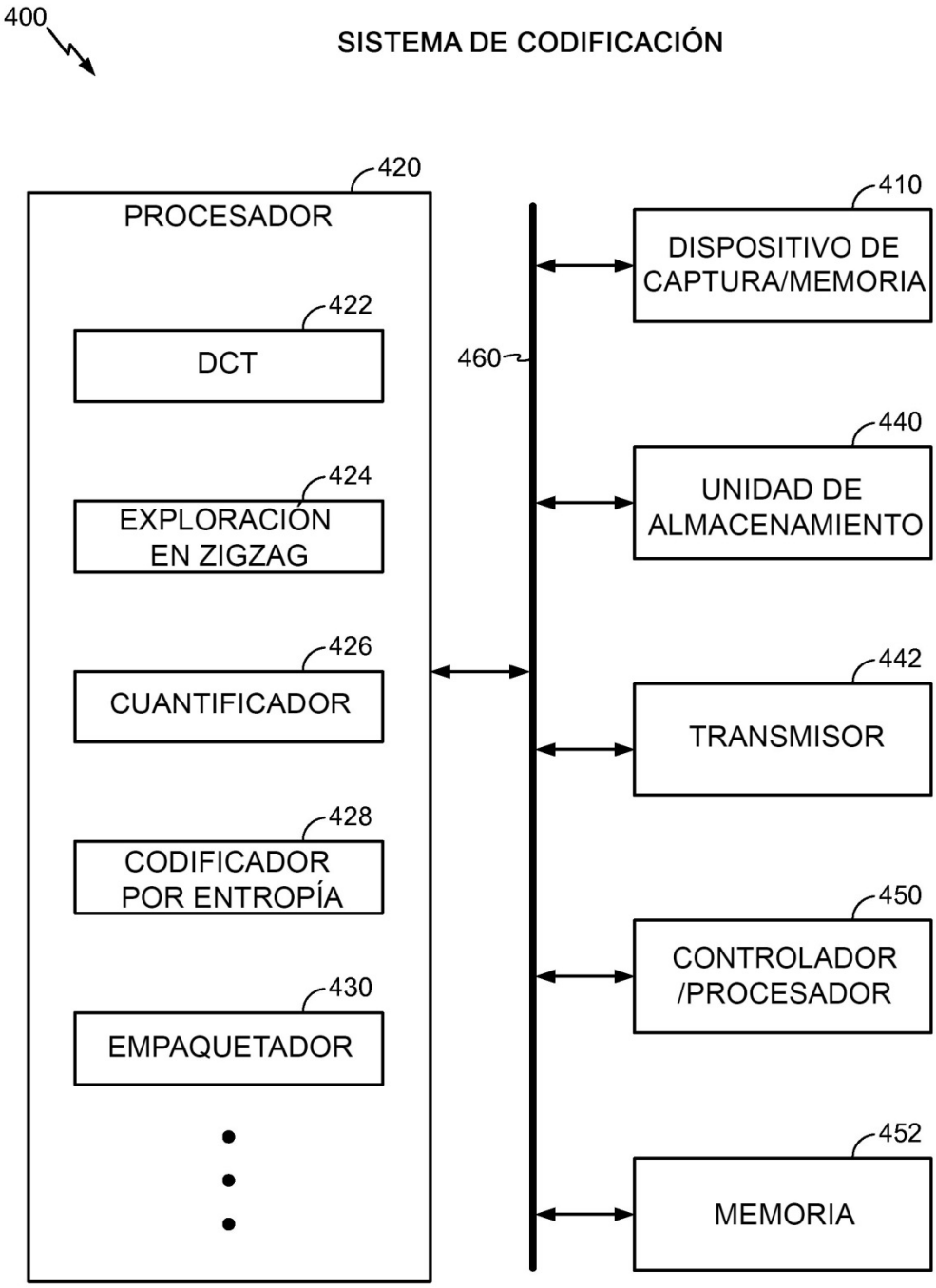
400
SISTEMA DE CODIFICACIÓN
420
PROCESADOR
422
DCT
424
EXPLORACIÓN EN ZIGZAG
426
CUANTIFICADOR
428
CODIFICADOR POR ENTROPÍA
430
EMPAQUETADOR
460
410
DISPOSITIVO DE CAPTURA/MEMORIA
440
UNIDAD DE ALMACENAMIENTO
442
TRANSMISOR
450
CONTROLADOR /PROCESADOR
452
MEMORIA

FIG. 4

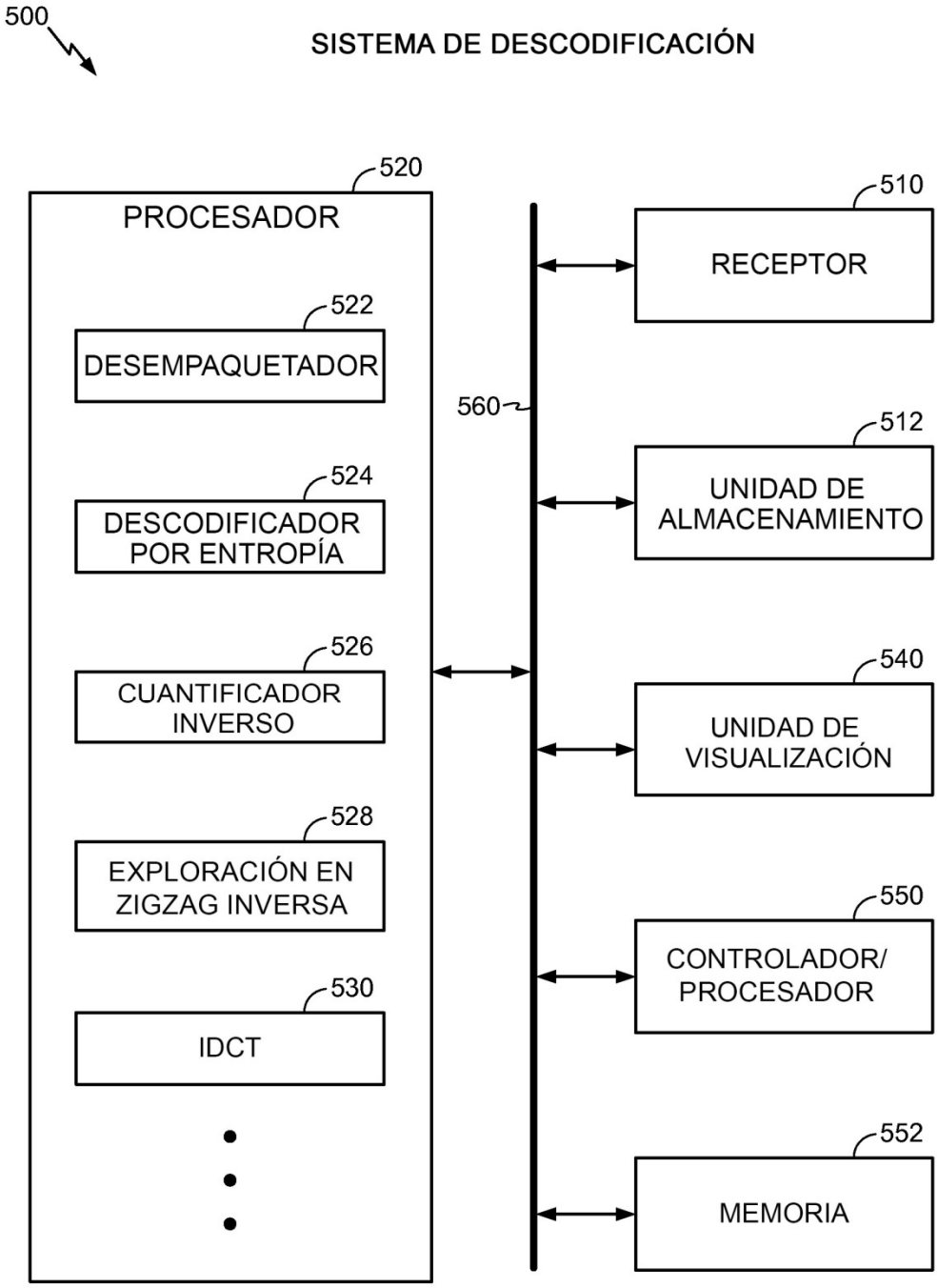
500
SISTEMA DE DESCODIFICACIÓN
520
PROCESADOR
522
DESEMPAQUETADOR
524
DESCODIFICADOR POR ENTROPÍA
526
CUANTIFICADOR INVERSO
528
EXPLORACIÓN EN ZIGZAG INVERSA
530
IDCT
560
510
RECEPTOR
512
UNIDAD DE ALMACENAMIENTO
540
UNIDAD DE VISUALIZACIÓN
550
CONTROLADOR/ PROCESADOR
552
MEMORIA

FIG. 5