

(19)日本国特許庁(JP)

(12)公開特許公報(A)

(11)公開番号
特開2022-45909
(P2022-45909A)

(43)公開日 令和4年3月22日(2022.3.22)

(51)国際特許分類	F I	テーマコード (参考)
H 0 4 N 19/13 (2014.01)	H 0 4 N 19/13	5 C 1 5 9
H 0 4 N 19/136 (2014.01)	H 0 4 N 19/136	
H 0 4 N 19/176 (2014.01)	H 0 4 N 19/176	
H 0 4 N 19/70 (2014.01)	H 0 4 N 19/70	

審査請求 未請求 請求項の数 12 O L (全55頁)

(21)出願番号 特願2021-139492(P2021-139492)	(71)出願人 000005049
(22)出願日 令和3年8月30日(2021.8.30)	シャープ株式会社
(31)優先権主張番号 特願2020-150920(P2020-150920)	大阪府堺市堺区匠町1番地
(32)優先日 令和2年9月9日(2020.9.9)	(74)代理人 100160783
(33)優先権主張国・地域又は機関 日本国(JP)	弁理士 堅田 裕之
(31)優先権主張番号 特願2020-160471(P2020-160471)	(72)発明者 橋本 知典
(32)優先日 令和2年9月25日(2020.9.25)	大阪府堺市堺区匠町1番地 シャープ株
(33)優先権主張国・地域又は機関 日本国(JP)	式会社内
(31)優先権主張番号 特願2020-169643(P2020-169643)	(72)発明者 猪飼 知宏
(32)優先日 令和2年10月7日(2020.10.7)	大阪府堺市堺区匠町1番地 シャープ株
(33)優先権主張国・地域又は機関 日本国(JP)	式会社内
(31)優先権主張番号 特願2020-195609(P2020-195609)	(72)発明者 青野 友子
最終頁に続く	大阪府堺市堺区匠町1番地 シャープ株
	式会社内
	(72)発明者 八杉 将伸
	最終頁に続く

(54)【発明の名称】 画像復号装置及び画像符号化装置

(57)【要約】

【課題】変換係数の絶対値が大きい場合でも、絶対値に応じた適切なライスパラメータを導出することにより、符号化効率が向上する効果を奏する画像復号装置及び画像符号化装置を提供する。

【解決手段】変換係数の絶対値を表すシンタックス要素を復号するTU復号部は、ゴロム・ライス符号によって前記シンタックス要素を復号し、復号された周囲の変換係数の絶対値和を重み付けし、補正項およびTU単位で導出される状態変数を重み付けした項を加えた値を、シフト値でシフトをすることでライスパラメータを導出し、前記重み、補正項、TU単位で導出される状態変数に係る重み、シフト値はブロックサイズに応じて異なることを特徴とする。

【選択図】図17

図17

(a)
$$cRiceParam = floor(\log2((parA * locSumAbs + parB) \gg parC)) + parE$$

(b)
$$cRiceParam = floor(\log2((parA * locSumAbs + parB)) + parE$$

(c)
$$cRiceParam = floor(\log2((locSumAbs \ll parS) + parB)) + parE$$

(d)
$$cRiceParam = floor(\log2((parA * locSumAbs + parB + parD * remstat) \gg parC)) + parE$$

(e)
$$cRiceParam = floor(\log2((parA * locSumAbs + parB + parD * remstat)) + parE$$

(f)
$$cRiceParam = floor(\log2((locSumAbs \ll parS) + parB + (remstat \ll parT)) + parE$$

【特許請求の範囲】

【請求項 1】

符号化データから、変換係数の絶対値を表すシンタックス要素abs_remainderとdec_abs_levelを復号するTU復号部を備え、

前記TU復号部は、abs_remainderとdec_abs_levelを、ライスパラメータを用いたゴロム・ライス符号によって復号することを特徴とする画像復号装置。

【請求項 2】

前記TU復号部は、

復号した周囲の変換係数の絶対値和を重みにより重み付けし、補正項を加えた値を用いて上記ライスパラメータを導出することを特徴とする請求項 1 に記載の画像復号装置。

10

【請求項 3】

前記重みおよび補正項はブロックサイズに応じて異なることを特徴とする請求項 2 に記載の画像復号装置。

【請求項 4】

前記TU復号部は、

復号した周囲の変換係数の絶対値和と重みとの積と、変換係数の大きさから導出される状態変数と重みとの積と、上記補正項との和の 2 を底とする対数値を用いて、上記ライスパラメータを導出することを特徴とする請求項 1 に記載の画像復号装置。

【請求項 5】

前記重み、補正項、状態変数に係る重みはブロックサイズに応じて異なることを特徴とする請求項 4 に記載の画像復号装置。

20

【請求項 6】

変換係数の絶対値を表すシンタックス要素abs_remainderとdec_abs_levelを符号化するTU符号化部を備え、

前記TU符号化部は、ライスパラメータを用いたゴロム・ライス符号によってabs_remainderとdec_abs_levelを、符号化することを特徴とする画像符号化装置。

【請求項 7】

前記TU符号化部は、

復号の周囲の変換係数の絶対値和を重みにより重み付けし、補正項および状態変数を重み付けした項を加えた値を、シフト値でシフトした値から、上記ライスパラメータを導出することを特徴とする請求項 6 に記載の画像符号化装置。

30

【請求項 8】

前記重みおよび補正項はブロックサイズに応じて異なることを特徴とする請求項 7 に記載の画像符号化装置。

【請求項 9】

前記TU符号化部は、

復号した周囲の変換係数の絶対値和を重みにより重み付けし、補正項および状態変数を重み付けした項を加えた値を、シフト値でシフトした値から、上記ライスパラメータを導出することを特徴とする請求項 6 に記載の画像符号化装置。

【請求項 10】

前記重み、補正項、状態変数に係る重みはブロックサイズに応じて異なることを特徴とする請求項 9 に記載の画像符号化装置。

40

【請求項 11】

前記TU復号部は、復号した周囲の変換係数の絶対値和をシフトし、補正項を加えた値を用いてライスパラメータを導出することを特徴とする請求項 1 に記載の画像復号装置。

【請求項 12】

前記TU復号部は、復号した周囲の変換係数の絶対値和をシフトし、補正項を加えた値の 2 を底とする対数を用いてライスパラメータを導出することを特徴とする請求項 11 に記載の画像復号装置。

【発明の詳細な説明】

50

【技術分野】

【0001】

本発明の実施形態は、画像復号装置及び画像符号化装置に関する。

【背景技術】

【0002】

画像を効率的に伝送又は記録するために、画像を符号化することによって符号化データを生成する画像符号化装置、及び、当該符号化データを復号することによって復号画像を生成する画像復号装置が用いられている。

【0003】

具体的な画像符号化方式としては、例えば、H.264/AVCやHEVC (High-Efficiency Video Coding) 方式等が挙げられる。

10

【0004】

このような画像符号化方式においては、動画画像を構成する画像（ピクチャ）は、画像を分割することにより得られるスライス、スライスを分割することにより得られる符号化ツリーユニット（CTU：Coding Tree Unit）、符号化ツリーユニットを分割することで得られる符号化単位（符号化ユニット（Coding Unit：CU）と呼ばれることもある）、及び、符号化単位を分割することにより得られる変換ユニット（TU：Transform Unit）からなる階層構造により管理され、CU毎に符号化／復号される。

【0005】

また、このような画像符号化方式においては、通常、入力画像を符号化／復号することによって得られる局所復号画像に基づいて予測画像が生成され、当該予測画像を入力画像（原画像）から減算して得られる予測誤差（「差分画像」又は「残差画像」と呼ぶこともある）が符号化される。予測画像の生成方法としては、画面間予測（インター予測）、及び、画面内予測（イントラ予測）が挙げられる。

20

【0006】

また、近年の画像符号化及び復号技術として非特許文献1が挙げられる。非特許文献1には、符号化効率を高めるため、変換係数の絶対値を表すシンタックス要素を復号する際、変換係数の絶対値はゴロム・ライス符号によりバイナリ化され、バイパス符号で復号される。シンタックス要素は、例えばabs_remainder、dec_abs_levelである。ゴロム・ライス符号は、周囲の係数の絶対値を参照して導出されたライスパラメータを用いて導出される。特に、周囲の係数の絶対値の和を求め、0～31にクリップした値を用いて、テーブルからライスパラメータを導出する、という方式によりこれを実現している。

30

【先行技術文献】

【非特許文献】

【0007】

【非特許文献1】"Versatile Video Coding (Draft 10)", JVET-S2001, Joint Video Exploration Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11

【発明の概要】

【発明が解決しようとする課題】

40

【0008】

非特許文献1の技術では、変換係数の絶対値が大きく、かつ、ライスパラメータの値が小さい場合、ゴロム・ライス符号によるバイナリ化によって符号量が大きくなり、符号化効率が低下する場合がある。特に、プロフェッショナル向けの映像では、量子化幅が小さいため係数の絶対値が大きくなりやすい。このような場合、非特許文献1のテーブルを用いてライスパラメータを導出すると、ライスパラメータ値に上限があるため、変換係数値が大きいと飽和して符号化効率が低下するという問題がある。

【課題を解決するための手段】

【0009】

上記課題を解決するために、本発明の一態様に係る画像復号装置は、符号化データから、

50

変換係数の絶対値を表すシンタックス要素 (abs_remainder、dec_abs_level) を復号するTU復号部を備え、TU復号部は、abs_remainderとdec_abs_levelを、ライスパラメータを用いたゴロム・ライス符号によって復号することの特徴とする。

【0010】

前記TU復号部は、復号した周囲の変換係数の絶対値和を重みにより重み付けし、補正項および状態変数を重み付けした項を加えた値を、シフト値でシフトした値から、ライスパラメータを導出することの特徴とする。

【0011】

前記重み、補正項、状態変数に係る重み、はブロックサイズに応じて異なることを特徴とする。

【0012】

このような構成をとることで、変換係数の絶対値が大きい場合でも、変換係数の絶対値に応じた適切なライスパラメータを導出することにより、符号化効率が向上する効果を奏する。

【発明の効果】

【0013】

以上の構成によれば、上記問題の何れかの解決を図ることができる。

【図面の簡単な説明】

【0014】

【図1】本実施形態に係る画像伝送システムの構成を示す概略図である。

【図2】本実施形態に係る動画像符号化装置を搭載した送信装置、及び、動画像復号装置を搭載した受信装置の構成について示した図である。PROD_Aは動画像符号化装置を搭載した送信装置を示しており、PROD_Bは動画像復号装置を搭載した受信装置を示している。

【図3】本実施形態に係る動画像符号化装置を搭載した記録装置、及び、動画像復号装置を搭載した再生装置の構成について示した図である。PROD_Cは動画像符号化装置を搭載した記録装置を示しており、PROD_Dは動画像復号装置を搭載した再生装置を示している。

【図4】符号化ストリームのデータの階層構造を示す図である。

【図5】CTUの分割例を示す図である。

【図6】動画像復号装置の構成を示す概略図である。

【図7】動画像復号装置の概略的動作を説明するフローチャートである。

【図8】TU復号部の構成を示す概略図である。

【図9】量子化変換係数に関するシンタックス要素を示す図である。

【図10】量子化変換係数に関するシンタックス要素を示す図である。

【図11a】RRCに関するシンタックス表である。

【図11b】RRCに関するシンタックス表である。

【図11c】RRCに関するシンタックス表である。

【図12a】TSRCに関するシンタックス表である。

【図12b】TSRCに関するシンタックス表である。

【図13】動画像符号化装置の構成を示すブロック図である。

【図14】ライスパラメータ導出に関する図である。

【図15】本実施形態に係るブロックインデックスを用いたa,b,cの導出表である

【図16】本実施形態に係る中間推定値valからライスパラメータを導出する例である。

【図17】本実施形態に係るlocSumAbs(=locSumAbs0)からライスパラメータを導出する例である。

【図18】本実施形態に係るライスパラメータの予測係数のシンタックステーブルの例である。

【発明を実施するための形態】

【0015】

10

20

30

40

50

〔実施形態1〕

以下、図面を参照しながら本発明の実施形態について説明する。

【0016】

図1は、本実施形態に係る画像伝送システム1の構成を示す概略図である。

【0017】

画像伝送システム1は、符号化対象画像を符号化した符号化ストリームを送信し、送信された符号化ストリームを復号し画像を表示するシステムである。画像伝送システム1は、動画像符号化装置（画像符号化装置）11、ネットワーク21、動画像復号装置（画像復号装置）31、及び画像表示装置（画像表示装置）41を含んで構成される。

【0018】

動画像符号化装置11には画像Tが入力される。

【0019】

ネットワーク21は、動画像符号化装置11が生成した符号化ストリームTeを動画像復号装置31に伝送する。ネットワーク21は、インターネット（Internet）、広域ネットワーク（WAN:Wide Area Network）、小規模ネットワーク（LAN:Local Area Network）又はこれらの組み合わせである。ネットワーク21は、必ずしも双方向の通信網に限らず、地上デジタル放送、衛星放送等の放送波を送信する一方向の通信網であってもよい。また、ネットワーク21は、DVD（Digital Versatile Disc:登録商標）、BD（Blue-ray Disc:登録商標）等の符号化ストリームTeを記録した記憶媒体で代替されてもよい。

【0020】

動画像復号装置31は、ネットワーク21が伝送した符号化ストリームTeのそれぞれを復号し、復号した1又は複数の復号画像Tdを生成する。

【0021】

画像表示装置41は、動画像復号装置31が生成した1又は複数の復号画像Tdの全部又は一部を表示する。画像表示装置41は、例えば、液晶ディスプレイ、有機EL（Electroluminescence）ディスプレイ等の表示デバイスを備える。ディスプレイの形態としては、据え置き、モバイル、HMD等が挙げられる。また、動画像復号装置31が高い処理能力を有する場合には、画質の高い画像を表示し、より低い処理能力しか有しない場合には、高い処理能力、表示能力を必要としない画像を表示する。

【0022】

<演算子>

本明細書で用いる演算子を以下に記載する。

【0023】

>>は右ビットシフト、<<は左ビットシフト、&はビットワイズAND、|はビットワイズOR、|=はOR代入演算子であり、||は論理和を示す。

【0024】

x?y:zは、xが真（0以外）の場合にy、xが偽（0）の場合にzをとる3項演算子である。

【0025】

Clip3(a,b,c)は、cをa以上b以下の値にクリップする関数であり、c<aの場合にはaを返し、c>bの場合にはbを返し、その他の場合にはcを返す関数である（ただし、a<=b）。

【0026】

abs(a)はaの絶対値を返す関数である。

【0027】

Int(a)はaの整数値を返す関数である。

【0028】

floor(a)はa以下の最小の整数を返す関数である。

【0029】

ceil(a)はa以上の最大の整数を返す関数である。

【0030】

a/d は d による a の除算（小数点以下切り捨て）を表す。

【0031】

＜符号化ストリームTeの構造＞

本実施形態に係る動画像符号化装置11及び動画像復号装置31の詳細な説明に先立って、動画像符号化装置11によって生成され、動画像復号装置31によって復号される符号化ストリームTeのデータ構造について説明する。

【0032】

図4は、符号化ストリームTeにおけるデータの階層構造を示す図である。符号化ストリームTeは、例示的に、シーケンス、及びシーケンスを構成する複数のピクチャを含む。図4には、それぞれ、シーケンスSEQを既定する符号化ビデオシーケンス、ピクチャPICTを規定する符号化ピクチャ、スライスSを規定する符号化スライス、スライスデータを規定する符号化スライスデータ、符号化スライスデータに含まれる符号化ツリーユニット、符号化ツリーユニットに含まれる符号化ユニットを示す図が示されている。

10

【0033】

（符号化ビデオシーケンス）

符号化ビデオシーケンスでは、処理対象のシーケンスSEQを復号するために動画像復号装置31が参照するデータの集合が規定されている。シーケンスSEQは、図4の符号化ビデオシーケンスに示すように、ビデオパラメータセット（Video Parameter Set）、シーケンスパラメータセットSPS（Sequence Parameter Set）、ピクチャパラメータセットPPS（Picture Parameter Set）、ピクチャヘッダ（picture header）、ピクチャPICT、及び、付加拡張情報SEI（Supplemental Enhancement Information）を含んでいる。

20

【0034】

ビデオパラメータセットVPSは、複数のレイヤから構成されている画像において、複数の画像に共通する符号化パラメータの集合及び画像に含まれる複数のレイヤ及び個々のレイヤに関連する符号化パラメータの集合が規定されている。

【0035】

シーケンスパラメータセットSPSでは、対象シーケンスを復号するために動画像復号装置31が参照する符号化パラメータの集合が規定されている。例えば、ピクチャの幅や高さが規定される。なお、SPSは複数存在してもよい。その場合、PPSから複数のSPSの何れかを選択する。

30

【0036】

ピクチャパラメータセットPPSでは、対象シーケンス内の各ピクチャを復号するために動画像復号装置31が参照する符号化パラメータの集合が規定されている。例えば、ピクチャの復号に用いられる量子化幅の基準値（pic_init_qp_minus26）や重み付き予測の適用を示すフラグ（weighted_pred_flag）及びスケーリングリスト（量子化マトリックス）が含まれる。なお、PPSは複数存在してもよい。その場合、対象シーケンス内の各ピクチャから複数のPPSの何れかを選択する。

【0037】

ピクチャヘッダでは、1枚の符号化ピクチャに含まれる全てのスライスに共通の符号化パラメータが定義される。例えばPOC（Picture Order Count）や分割に関する符号化パラメータが含まれる。

40

【0038】

（符号化ピクチャ）

符号化ピクチャでは、処理対象のピクチャPICTを復号するために動画像復号装置31が参照するデータの集合が規定されている。ピクチャPICTは、図4の符号化ピクチャに示すように、スライス0～スライスNS-1を含む（NSはピクチャPICTに含まれるスライスの総数）。

【0039】

50

なお、以下、スライス0～スライスNS-1のそれぞれを区別する必要が無い場合、符号の添え字を省略して記述することがある。また、以下に説明する符号化ストリームTeに含まれるデータであって、添え字を付している他のデータについても同様である。

【0040】

(符号化スライス)

符号化スライスでは、処理対象のスライスSを復号するために動画像復号装置31が参照するデータの集合が規定されている。スライスは、図4の符号化スライスに示すように、スライスヘッダ、及び、スライスデータを含んでいる。

【0041】

スライスヘッダには、対象スライスの復号方法を決定するために動画像復号装置31が参照する符号化パラメータ群が含まれる。スライスタイプを指定するスライスタイプ指定情報(slice_type)は、スライスヘッダに含まれる符号化パラメータの一例である。

【0042】

スライスタイプ指定情報により指定可能なスライスタイプとしては、(1)符号化の際にイントラ予測のみを用いるIスライス、(2)符号化の際に単方向予測、又は、イントラ予測を用いるPスライス、(3)符号化の際に単方向予測、双方向予測、又は、イントラ予測を用いるBスライス等が挙げられる。なお、インター予測は、単予測、双予測に限定されず、より多くの参照ピクチャを用いて予測画像を生成してもよい。以下、P、Bスライスと呼ぶ場合には、インター予測を用いることができるブロックを含むスライスを指す。

【0043】

なお、スライスヘッダは、ピクチャパラメータセットPPSへの参照(pic_parameter_set_id)を含んでもよい。

【0044】

(符号化スライスデータ)

符号化スライスデータでは、処理対象のスライスデータを復号するために動画像復号装置31が参照するデータの集合が規定されている。スライスデータは、図4の符号化スライスヘッダに示すように、CTUを含んでいる。CTUは、スライスを構成する固定サイズ(例えば64x64)のブロックであり、最大符号化単位(LCU:Largest Coding Unit)と呼ぶこともある。

【0045】

(符号化ツリーユニット)

図4の符号化ツリーユニットには、処理対象のCTUを復号するために動画像復号装置31が参照するデータの集合が規定されている。CTUは、再帰的な4分木分割(QT(Quad Tree)分割)、2分木分割(BT(Binary Tree)分割)あるいは3分木分割(TT(Ternary Tree)分割)により符号化処理の基本的な単位である符号化ユニットCUに分割される。BT分割とTT分割を合わせてマルチツリー分割(MT(Multi Tree)分割)と呼ぶ。再帰的な4分木分割により得られる木構造のノードのことを符号化ノード(Coding Node)と称する。4分木、2分木、及び3分木の間ノードは、符号化ノードであり、CTU自身も最上位の符号化ノードとして規定される。

【0046】

CTは、CT情報として、QT分割を行うか否かを示す分割フラグを含む。図5に分割の例を示す。

【0047】

CUは符号化ノードの末端ノードであり、これ以上分割されない。CUは符号化処理の基本的な単位となる。

【0048】

(符号化ユニット)

図4の符号化ユニットに示すように、処理対象の符号化ユニットを復号するために動画像復号装置31が参照するデータの集合が規定されている。具体的には、CUは、CUヘッダC

10

20

30

40

50

UH、予測パラメータ、変換パラメータ、量子化変換係数等から構成される。CUヘッダでは予測モード等が規定される。

【0049】

予測処理は、CU単位で行われる場合と、CUをさらに分割したサブCU単位で行われる場合がある。CUとサブCUのサイズが等しい場合には、CU中のサブCUは1つである。CUがサブCUのサイズよりも大きい場合、CUは、サブCUに分割される。例えばCUが8x8、サブCUが4x4の場合、CUは水平2分割、垂直2分割からなる、4つのサブCUに分割される。

【0050】

予測の種類（予測モードCuPredMode）は、イントラ予測(MODE_INTRA)と、インター予測(MODE_INTER)の2つを少なくとも備える。さらにイントラブロックコピー予測(MODE_IBC)を備えても良い。イントラ予測、イントラブロックコピー予測は、同一ピクチャ内の予測であり、インター予測は、互いに異なるピクチャ間（例えば、表示時刻間、レイヤ画像間）で行われる予測処理を指す。

【0051】

変換・量子化処理はCU単位で行われるが、量子化変換係数は4x4等のサブブロック単位でエントロピー符号化してもよい。

【0052】

予測画像は、ブロックに付随する予測パラメータによって導出される。予測パラメータには、イントラ予測とインター予測の予測パラメータがある。

【0053】

（動画像復号装置の構成）

本実施形態に係る動画像復号装置31（図6）の構成について説明する。

【0054】

動画像復号装置31は、エントロピー復号部301、パラメータ復号部（予測画像復号装置）302、ループフィルタ305、参照ピクチャメモリ306、予測パラメータメモリ307、予測画像生成部（予測画像生成装置）308、逆量子化・逆変換部311、及び加算部312を含んで構成される。なお、後述の動画像符号化装置11に合わせ、動画像復号装置31にループフィルタ305が含まれない構成もある。

【0055】

パラメータ復号部302は、さらに、ヘッダ復号部3020、CT情報復号部3021、及びCU復号部3022（予測モード復号部）を備えており、CU復号部3022はTU復号部3024を備えている。これらを総称して復号モジュールと呼んでもよい。ヘッダ復号部3020は、符号化データからVPS、SPS、PPS等のパラメータセット情報、スライスヘッダ（スライス情報）を復号する。CT情報復号部3021は、符号化データからCTを復号する。CU復号部3022は符号化データからCUを復号する。TU復号部3024は、符号化データから、変換スキップフラグtransform_skip_flagとQP更新情報（量子化補正值）と量子化変換係数(residual_coding)を復号する。

【0056】

TU復号部3024は、変換ユニットのサイズ(tbWidthとtbHeight)が所定の最大サイズ以下である場合(tbWidth ≤ MaxTsSize && tbHeight ≤ MaxTsSize)、transform_skip_flag[x0][y0][cIdx]を復号する。

【0057】

TU復号部3024は、TUに予測誤差が含まれている場合（例えば、tu_cbf_luma[x0][y0]が1の場合）、符号化データからQP更新情報と量子化変換係数を復号する。量子化変換係数の導出では複数のモード（例えばRRCモードとTSRCモード）を備えてもよい。具体的には、変換を用いる通常の予測誤差の導出(RRC:Regular Residual Coding)と、変換を用いない変換スキップモードでの予測誤差の導出(TSRC:Transform Skip Residual Coding)とで、異なる処理を実施してもよい。QP更新情報は、量子化パラメータQPの予測値である量子化パラメータ予測値qPpredからの差分値である。

【0058】

10

20

30

40

50

また、以降では処理の単位としてCTU、CUを使用した例を記載するが、この例に限らず、サブCU単位で処理をしてもよい。あるいはCTU、CUをブロック、サブCUをサブブロックと読み替え、ブロックあるいはサブブロック単位の処理としてもよい。

【0059】

エントロピー復号部301は、外部から入力された符号化ストリームTeに対してエントロピー復号を行って、個々の符号（シンタックス要素）をパースする。エントロピー符号化には、シンタックス要素の種類や周囲の状況に応じて適応的に選択したコンテキスト（確率モデル）を用いてシンタックス要素を可変長符号化する方式と、予め定められた表、あるいは計算式を用いてシンタックス要素を可変長符号化する方式がある。前者の例としてCABAC（Context Adaptive Binary Arithmetic Coding）が挙げられる。パースされた符号には、予測画像を生成するための予測情報及び、差分画像を生成するための予測誤差等がある。

10

【0060】

エントロピー復号部301は、分離した符号をパラメータ復号部302に出力する。分離した符号とは、例えば、予測モードCuPredModeである。どの符号を復号するかはの制御は、パラメータ復号部302の指示に基づいて行われる。

【0061】

（基本フロー）

図7は動画画像復号装置31の概略的動作を説明するフローチャートである。

20

【0062】

（S1100：パラメータセット情報復号）ヘッダ復号部3020は、符号化データからVPS、SPS、PPS、SEI、PH等のパラメータセット情報を復号する。

【0063】

（S1200：スライス情報復号）ヘッダ復号部3020は、符号化データからスライスヘッダ（スライス情報）を復号する。

【0064】

以下、動画画像復号装置31は、対象ピクチャに含まれる各CTUについて、S1300からS5000の処理を繰り返すことにより各CTUの復号画像を導出する。

【0065】

（S1300：CTU情報復号）CT情報復号部3021は、符号化データからCTUを復号する。

30

【0066】

（S1400：CT情報復号）CT情報復号部3021は、符号化データからCTを復号する。

【0067】

（S1500：CU復号）CU復号部3022はS1510、S1520を実施して、符号化データからCUを復号する。また、CU復号部3022は、符号化データからCU単位で量子化パラメータの差分CuQpDeltaValを復号し、量子化パラメータを導出する。

【0068】

（S1510：CU情報復号）CU復号部3022は、符号化データからCU情報、予測情報、TU分割フラグsplit_transform_flag、CU残差フラグcbf_cb、cbf_cr、cbf_luma等を復号する。

40

【0069】

（S1520：TU情報復号）TU復号部3024は、TUに予測誤差が含まれている場合に、符号化データからQP更新情報と量子化変換係数を復号する。

【0070】

（S2000：予測画像生成）予測画像生成部308は、対象CUに含まれる各ブロックについて、予測情報に基づいて予測画像を生成する。

【0071】

（S3000：逆量子化・逆変換）逆量子化・逆変換部311は、対象CUに含まれる各TUについて、逆量子化・逆変換処理を実行する。

【0072】

50

(S4000:復号画像生成) 加算部312は、予測画像生成部308より供給される予測画像と、逆量子化・逆変換部311より供給される予測誤差とを加算することによって、対象CUの復号画像を生成する。

【0073】

(S5000:ループフィルタ) ループフィルタ305は、復号画像にデブロッキングフィルタ、SAO (Sample Adaptive Filter)、ALF (Adaptive Loop Filter)等のループフィルタをかけ、復号画像を生成する。

【0074】

(量子化変換係数の導出、残差符号化)

ロスレス符号化や原画の画素間相関が小さい時には、変換を実施しない方が、符号化効率
10
は高い場合がある。変換を実施しない技術を変換スキップ (Transform Skip) と称す。
変換スキップは、Identical Transformとも呼ばれ、量子化パラメータに応じた変換
係数のスケーリングのみを行う。変換スキップであるかはシンタックス要素transform_ skip_flagを用いて通知される。transform_skip_flagは、Y、Cb、Crの各色コンポー
ネント(cldx)ごとに通知されてもよい。

【0075】

変換を用いる通常の予測誤差の導出(RRC:Regular Residual Coding)と、変換スキップ
モードでの予測誤差の導出(TSRC:Transform Skip Residual Coding)とでは、予測
誤差の符号化方法も復号方法も異なる。

【0076】

図8はTU復号部3024のブロック図であり、RRC部30241とTSRC部30242が含まれる。
20
RRC部30241は変換を用いる通常の予測誤差を導出する処理部であり、TSRC部302
42は変換スキップモードでの予測誤差を導出する処理部である。

【0077】

図9(a)sps_transform_skip_enabled_flagは、各TUでtransform_skip_flagが通知
されるか否かを示すフラグである。sps_transform_skip_enabled_flag=1は、各
TUでtransform_skip_flagが通知されることを示す。sps_transform_skip_enabled_ flag=0は、各TUでtransform_skip_flagが通知されないことを示す。sps_tran
sform_skip_enabled_flagが通知されない場合、0と推定される。

【0078】

図10(b)のtransform_skip_flag[x0][y0][cldx]は、左上座標が(x0,y0)、色成分が
30
cldxのブロックにおいて、変換が適用されるか否かを示す。transform_skip_flag=1
(変換スキップモード)の場合、このブロックには変換は適用されない。transform_ s
kip_flag=0の場合、このブロックに変換が適用されるか否かは他のパラメータに依存す
る。

【0079】

図11と図12はRRCおよびTSRCでの変換係数(予測誤差)の符号化方法を示すシンタックス
である。

【0080】

(RRC部、RRCモード)

変換スキップしない通常の予測誤差符号化方法(図11)において、RRC部30241は、LAST
40
位置を示すシンタックス要素(図示しない)を復号し、LAST位置(LastSignificantCo
effX, LastSignificantCoeffY)を導出する。LAST位置は、TUの変換係数を低周波数
成分から高周波数成分の方向にスキャンする場合の最後の非ゼロ係数の位置である。高周
波数成分から順に変換係数(係数)を符号化、復号する場合、LAST位置は最初に復号す
る量子化変換係数の位置を示す。次に、RRC部30241は、LAST位置を参照してsb_coded_ flagを復号する。sb_coded_flagは、サブブロックに非ゼロ係数が含まれるか否か
を示すフラグである。サブブロックはTUを4x4単位に分割した領域である。sb_coded_ flag=1(サブブロックに非ゼロ係数が含まれる)であれば、RRC部30241はsig_coeff_ flagを復号する。sig_coeff_flagは係数値が非ゼロか否かを示すフラグである。sig_c
50

coeff_flag=1(係数値は非ゼロ)であれば、RRC部30241はabs_level_gtx_flag、par_level_flag、abs_remainder、dec_abs_levelを復号する。これらは係数の絶対値を示すシンタックス要素である。abs_level_gtx_flag[n][j]はスキャン位置nの係数の絶対値が(j < 1)+1より大きいかなを示すフラグである。abs_level_gtx_flag[n][j]が通知されない場合、0と推定される。par_level_flag[n]はスキャン位置nの係数のパリティである。par_level_flag[n][j]が通知されない場合、0と推定される。abs_remainder[n]はスキャン位置nの係数の絶対値の残り(remaining)を表し、ゴロム・ライス符号(Golomb-Rice code)により復号される。abs_remainder[n]が通知されない場合、0と推定される。dec_abs_level[n]はスキャン位置nにおける係数の絶対値を導出するための残差の絶対値である。dec_abs_level[n]が通知されない場合、0と推定される。RRC部30241はこれらのシンタックス要素から係数の絶対値を導出する。

10

【0081】

上記のように、RRC部30241は、TUのサブブロックにおいてLAST位置を復号することが特徴である。

【0082】

(TSRC部、TSRCモード)

変換を実施しない場合(変換スキップモード)の予測誤差符号化方法(図12)において、TSRC部30242は、各サブブロックのsb_coded_flagを復号する。sb_coded_flag=1(サブブロックに非ゼロ係数が含まれる)であれば、TSRC部30242は、サブブロック内の位置(xC, xC)の変換係数のsig_coeff_flag[xC][yC]を復号する。sig_coeff_flag=1(係数値は非ゼロ)であれば、TSRC部30242はcoeff_sign_flag、abs_level_gtx_flag、par_level_flag、abs_remainderを復号する。これらは係数の絶対値を示すシンタックス要素であり、定義は上述の通りである。TSRC部30242はこれらのシンタックス要素から係数の絶対値を導出する。

20

【0083】

上記のように、TSRC部30242は、TUのサブブロックにおいてLAST位置を復号しないことが特徴である。

【0084】

(abs_remainderとdec_abs_levelの復号)

TU復号部3024は、符号化データから、ライスパラメータcRiceParamを用いて、残差絶対値のシンタックス値(abs_remainderもしくはdec_abs_level、以下残差と記す)を復号する。当該シンタックスのバイナリゼーションには、プリフィックスとサフィックスとから構成される符号を用いる。プリフィックスとサフィックスとして、ゴロムライス符号のアルファ符号と固定長バイナリゼーションを用いてもよい。サフィックスはEG(k)(Exponential-Golomb code)符号を用いてもよい。なお、ゴロムライス符号では、プリフィックスをprefixValTRとsuffixValTRに分けて符号化し、EG(k)符号では、サフィックスをexp部のexpValとescape部のescapeValに分けて符号化する。残差(abs_remainderもしくはdec_abs_level)を $1 \ll cRiceParam$ 、 $((1 \ll expVal) - 1) \ll cRiceParam$ で割った商は各々prefixValTRとexpValを用いて符号化される。残差(abs_remainderもしくはdec_abs_level)を $1 \ll cRiceParam$ 、 $(1 \ll expVal \ll cRiceParam)$ で割った余りは、各々suffixValTR、escapeValを用いて符号化される。つまり、残差は以下の様に符号化される。

30

40

【0085】

$((prefixValTR + ((1 \ll expVal) - 1)) \ll cRiceParam) + suffixValTR + escapeVal$

残差が所定の値cMax以下の場合には、prefixValTRとsuffixValTRだけが符号化される。

【0086】

$prefixVal = (prefixValTR \ll cRiceParam) + suffixValTR$

残差が所定の値より大きい場合(suffixValが存在する場合)、cMaxと等しいprefixVal

50

を示すprefixValTRに加えてexpValとescapeValが符号化される。

【0087】

prefixVal = (prefixValTR < cRiceParam) ? cMax

suffixVal = (((1 < expVal) - 1) < cRiceParam) + escapeVal

ここでは残差は以下で表せる。プリフィックスだけで表せる場合の残差は下式である。

【0088】

残差 = ((prefixValTR) < cRiceParam) + suffixValTR

それ以外（プリフィックスだけで表せない場合、もしくは、prefixValTR=6）の場合の残差は下式である。

【0089】

残差 = ((prefixValTR + (1 < expVal) - 1) < cRiceParam) + escapeVal

また以下のように、prefixVal、suffixValを定義してもよい。

【0090】

prefixVal = (prefixValTR < cRiceParam) + suffixValTR

suffixVal = (((2 < expVal) - 2) < cRiceParam) + escapeVal

（流れ）

TU復号部3024は、まずプリフィックスprefixValの最大値cMaxを、cRiceParamから導出する。

【0091】

cMax = 6 < cRiceParam

TU復号部3024は、後述のTruncated Rice (TR)のバイナリゼーションに基づいて符号化データからプリフィックスの値 (prefixVal) を復号する。なお、プリフィックスと残差の関係は以下のとおりである。

【0092】

prefixVal = Min(cMax, abs_remainder[n])

あるいは以下でもよい。

【0093】

prefixVal = Min(cMax, dec_abs_level)

続いて、残差がcMaxより大きい場合（prefixValだけでは残差を表現できない場合）、TU復号部3024は、以下の関係式を満たすsuffixValをさらに復号する。

【0094】

suffixVal = abs_remainder[n] - cMax

あるいは以下でもよい。

【0095】

suffixVal = dec_abs_level - cMax

具体的には、TU復号部3024は、suffixValが存在する場合（TRのprefixValのビット列が長さ6の1の列の"111111"の場合）、符号化データからLimited k-th order Exp-Golomb (EG(k))のバイナリゼーションに基づいてsuffixValを復号する。ここでのLimited k-th order Exp-Golombは、次数kをcRiceParam、変数maxPreExtLenを11、変数truncSuffixLenを15とする。maxPreExtLenは固定値(11)とせず、変換係数のレンジlog2TransformRangeに応じて導出してもよい。例えば、以下の式で導出してもよい。

【0096】

maxPreExtLen = (32 - 6) - log2TransformRange = 26 - log2TransformRange

例えば、log2TransformRangeが後述の(式R-1)の場合、以下のように導出してもよい。

【0097】

maxPreExtLen = extended_precision_processing_flag ? 26 - Max(15, BitDepth + BDOFFSET) : 11

（Truncated Rice (TR)のバイナリゼーション）

10

20

30

40

50

TRバイナリゼーションの入力symbolValはprefixValであり、TRバイナリゼーションでは、symbolVal(==prefixVal)の $(1 \ll cRiceParam)$ による商をprefixValTR、余りをsuffixValTRとして符号化する。なお、TR符号の入力symbolValとprefixValTRの関係は以下のとおりである。

【0098】

prefixValTR = symbolVal $\gg$ cRiceParam

prefixValTRは以下の定義でビット列に変換される。prefixValTR $\ll$ (cMax $\gg$ cRiceParam)の場合、prefixValTR+1の長さのbinIdxでインデックスされたビット列である。binIdx $\ll$ prefixValTRの場合、bin(bins)は"1"、binIdx==prefixValTRの場合、binは"0"である。それ以外 (prefixValTR == (cMax $\gg$ cRiceParam))の場合、長さがcMax $\gg$ cRiceParamの"1"の列である。なお、binとは、"0"か"1"からなるビット列である。binIdxとは、ビット列の先頭からの位置を示す値であり、binIdx=0が先頭（バイナリ列の左端）に対応する。

10

以下にprefixValTRが0～5の場合のビット列 (bins) の例を示す。

【0099】

prefixValTR==0 bins=0

prefixValTR==1 bins=10

prefixValTR==2 bins=110

prefixValTR==3 bins=1110

prefixValTR==4 bins=11110

20

prefixValTR==5 bins=111110

...

ただし、symbolVal $\gg$ cMax(すなわちprefixVal==(6 $\ll$ cRiceParam))の場合は下式である。

【0100】

prefixValTR==6 bins=111111

TR符号のprefixValTRのbinsはtruncatedなアルファ符号 (unary符号) である。ここでtruncatedであるとは最大値に一致する場合には、最後の0を省略することを意味する。

【0101】

cMaxがsymbolValより大きく(symbolVal $\ll$ cMax)、cRiceParam $\gg$ 0の場合には、TRビン列のsuffixTRが存在し、下記のように導出される。suffixValTRはTRビット列のsuffixである。

30

【0102】

suffixValTR = symbolVal - (prefixValTR $\ll$ cRiceParam)

また、TRビット列のsuffixのバイナリゼーションは、最大値cMaxFLを $(1 \ll cRiceParam) - 1$ とする、固定長バイナリゼーション (後述) を用いる。

【0103】

なお、symbolVal==cMaxの場合には、残差のサフィックスは、上記TRバイナリゼーションのsuffixValTRではなく、EG(k)のバイナリゼーションを用いる。

40

【0104】

(Limited k-th order Exp-Golomb/EG(k)のバイナリゼーション)

残差のsuffixValはEG(k)のバイナリゼーションを用いてもよい。EG(k)は、次数k(=cRiceParam)とmaxPreExtLenを入力として得られるバイナリゼーションであり、以下の疑似コードで示す。EG(k)は長さmaxPreExtLen以下からなるexp部と、escapeLength長の固定長バイナリゼーションのescape部からなる。preExtLenは、exp部のput(1)の数をカウントする変数であり、exp部で符号化される値espValに等しい。つまりexpVal = preExtLen。対応する残差の値はexpVal $\ll$ cRiceParamである。escapeValはescape部で符号化される値である。

// exp部

50

```

codeValue = symbolVal > > k
preExtLen = 0
while((preExtLen < maxPreExtLen) && (codeValue > ((2 < <preExtLen) -
2))) {
preExtLen++
put(1)
}
if(preExtLen == maxPreExtLen)
escapeLength = truncSuffixLen
else {
escapeLength = preExtLen + k
put(0)
}

```

10

// escape部

```

symbolVal = symbolVal - (((1 < <preExtLen)-1) < <k)
while((escapeLength--)> 0)
put((symbolVal > > escapeLength) & 1)

```

ここで、symbolValは入力値であり、残差からprefixVal(==cMax)を引いた値である。put(X)は、X("0"または"1")をビット列の最後に追加してビット列を作る操作を示す。ビット列の初期値は空({})である。escapeLengthは、symbolValにおいてescape部から出力されるビット位置を示す変数である。

20

なお、escape部は以下のようにしてもよい。

```

【0105】
symbolVal = symbolVal - (((2 < <preExtLen)-2) < <k)
while((escapeLength--)> 0)
put((symbolVal > > escapeLength) & 1)

```

(固定長バイナリゼーション)

固定長バイナリゼーションは、最大値cMaxFL、値symbolValに対して、Ceil(Log2(cMaxFL+1))の長さfixedLengthのunsigned整数列(2進数列)を用いて行う。なお、位置binIdxの0が最上位ビット(MSB)に対応し、binIdxが大きくなるにつれて最下位ビット(LSB)に近づく。

30

```

【0106】

```

(残差の復号のまとめ)

TU復号部3024は、prefixValとsuffixValから構成される残差を復号する。

```

【0107】

```

TU復号部3024は、cRiceParamからcMaxを導出し、cMaxを上限として符号化データからprefixValを復号し、suffixValが存在する場合にはsuffixValを復号する。

```

【0108】

```

具体的には、TU復号部3024は、TRバイナリゼーションのprefixValTRを復号し、symbolVal < cMax、cRiceParam > 0 (つまりprefixVal < cMax、prefixVal < 6)の場合には、さらに符号化データから固定長バイナリゼーションのsuffixValTRを復号する。この場合、suffixValが存在しない。

40

```

【0109】

```

TU復号部3024は、上記suffixValが存在しない場合、以下の式で、abs_remainder[n]、dec_abs_levelを導出する。

```

【0110】

```

```

abs_remainder[n] = (prefixValTR < <cRiceParam) + suffixValTR
dec_abs_level[n] = (prefixValTR < <cRiceParam) + suffixValTR

```

それ以外の場合 (prefixVal == cMax、prefixValTR == 6の場合)、TU復号部3024は、EG(k)によりsuffixValを復号し下式で、abs_remainder[n]、dec_abs_level

50

[n]を導出する。

【0111】

$\text{abs_remainder}[n] = \text{cMax} + \text{suffixVal}$

$\text{dec_abs_level}[n] = \text{cMax} + \text{suffixVal}$

ここで、suffixValがEG(k)のexp部のpreExtLenとescape部のescapeValからなる場合、

$\text{suffixVal} = (\text{preExtLen} \ll \text{cRiceParam}) + \text{escapeVal}$

である。

【0112】

まとめると、TU復号部3024は、符号化データからTRバイナリゼーションのプリフィックスprefixValと、EG(k)のexp部のpreExtLenとescape部のescapeValからなるsuffixValを復号して、残差(abs_remainder[n]もしくはdec_abs_level[n])を導出する。

10

【0113】

$\text{abs_remainder}[n] = \text{prefixVal} + \text{suffixVal} = (\text{prefixValTR} \ll \text{cRiceParam}) + (\text{prefixValTR} \ll 6) ? \text{suffixValTR} : (1 \ll \text{expVal} \ll \text{cRiceParam}) + \text{escapeVal}$

ここで、(prefixValTR < 6)の場合にpreExtLen=escapeVal=0とすると、

$\text{abs_remainder}[n] = (\text{prefixValTR} + \text{preExtLen}) \ll \text{cRiceParam} + (\text{prefixValTR} \ll 6) ? \text{suffixValTR} : \text{escapeVal}$

20

dec_abs_level[n]の導出も同様である。

なお、truncatedなアルファ符号のprefixValTRのbinsとアルファ符号のpreExtLenのbinsを連結した符号もアルファ符号であるので、TRのprefixValとEG(k)のexp部は一体のものとして復号してもよい。この場合、最大長maxPreExtLen+6のアルファ符号を符号化データから読み取って、アルファ符号の長さprefixtmp(=prefixValTR+expVal)を復号する。ここで、maxPreExtLen+6は、prefixValTRの最大長6と、EG(k)のexp部の最大長maxPreExtLenの和である。

TU復号部3024は、 $1 \ll \text{cRiceParam}$ の商と余りに分けて、残差を以下のように導出してもよい。

【0114】

30

残差 = $(\text{prefixRes} \ll \text{cRiceParam}) + \text{suffixRes}$

復号されたアルファ符号の長さをprefixtmpとすると、残差の指数部prefixResは下式で表される。

【0115】

$\text{prefixRes} = (\text{prefixVal} \ll 6) ? \text{prefixValTR} : \text{prefixValTR} + (1 \ll \text{expVal}) - 1$

prefixtmp < 6の場合はprefixValTR=prefixtmp、prefixtmp >= 6の場合はprefixValTR = 6、expVal = prefixtmp-6であるので、以下で導出できる。

【0116】

prefixRes

= $(\text{prefixtmp} \ll 6) ? \text{prefixtmp} : 6 + (1 \ll (\text{prefixtmp} - 6)) - 1$

40

= $(\text{prefixtmp} \ll 6) ? \text{prefixtmp} : (1 \ll (\text{prefixtmp} - 6)) + 5$ また、

prefixRes = $(\text{prefixVal} \ll 6) ? \text{prefixValTR} : \text{prefixValTR} + (2 \ll \text{expVal}) - 2$ を用いる構成の場合には、

prefixRes

= $(\text{prefixtmp} \ll 6) ? \text{prefixtmp} : 6 + (2 \ll (\text{prefixtmp} - 6)) - 2$

= $(\text{prefixtmp} \ll 6) ? \text{prefixtmp} : (2 \ll (\text{prefixtmp} - 6)) + 4$

= $(\text{prefixtmp} \ll 6) ? \text{prefixtmp} : (1 \ll (\text{prefixtmp} - 5)) + 4$

ここでprefixtmp < 6はprefixtmp < 5としても処理できる。また、suffixValTRもescapeValもどちらも固定長バイナリゼーションなので、前者をcMaxFLのfixedLength、後者をescapeLength長のfixedLengthとして、同様にsuffixResも復号してよい。

50

【0117】

(量子化変換係数)

TU復号部3024は、位置nに対して、順に、符号化データからabs_remainderを復号し、量子化変換係数AbsLevelをAbsLevelPass1とabs_remainderから以下の式で導出する。nはサブブロック内のスキャン順で最後の係数位置以下で、所定のビット数を復号するまでの位置のいずれかである。

【0118】

$$\text{AbsLevel}[xC][yC] = \text{AbsLevelPass1}[xC][yC] + 2 * \text{abs_remainder}[n]$$

ここで、AbsLevelは、変換係数の絶対値（変換係数の差分の絶対値）である。AbsLevelPass1[xC][yC]は下記で表される。

【0119】

$$\text{AbsLevelPass1}[xC][yC] = \text{sig_coeff_flag}[xC][yC] + \text{par_level_flag}[n] + \text{abs_level_gtx_flag}[n][0] + 2 * \text{abs_level_gtx_flag}[n][1].$$

【0120】

TU復号部3024は、上記以外（所定のビット数を復号した時のn）の場合、符号化データからdec_abs_levelを復号し、以下の式によりAbsLevelを導出する。

【0121】

```
if (dec_abs_level[n] == ZeroPos[n])
  AbsLevel[xC][yC] = 0
else if (dec_abs_level[n] < ZeroPos)
  AbsLevel[xC][yC] = dec_abs_level[n] + 1
else /*dec_abs_level[n] > ZeroPos[n]*/
  AbsLevel[xC][yC] = dec_abs_level[n]
```

ここでZeroPosは以下の式で導出される。

【0122】

$$\text{ZeroPos}[n] = (\text{QState} \langle 2 ? 1 : 2 \rangle \langle \langle \text{cRiceParam} \rangle \rangle$$

ここでQStateは依存量子化の状態を示す。

【0123】

(ライスパラメータ導出)

係数の絶対値を表すシンタックス要素の内、abs_remainder、dec_abs_levelは、ゴロム・ライス符号(またはライス符号)でバイナリ化された値を、バイパス符号(CABACの等確率符号)により復号される。abs_remainderは変換係数の絶対値のオフセットに対する差分であり、dec_abs_levelは変換係数の絶対値である。

【0124】

なお、ゴロム・ライス符号は、シンタックス要素を前半部（プリフィックス）と後半（リマインダー、サフィックス）の2つの部分に分割して符号化する。プリフィックスには0に近い値を短い符号量で符号化可能なアルファ符号(unary符号)を用いる。サフィックスにはライスパラメータcRiceParamを用いた固定長バイナリゼーションもしくはEG(k)バイナリゼーションを用いる。ライスパラメータが大きいほど、小さな値のビット長はより長く、大きな値のビット長がより短くなる特性がある。シンタックス要素の値の大きさの発生確率に応じて、ライスパラメータを調整することで符号量の削減が可能である。

【0125】

本実施形態の動画像復号装置、動画像符号化装置では、既に導出された係数の絶対値から、復号するシンタックス要素の値を予測する。そして、その予測値から適切なライスパラメータを導出する。図14を用いて、以下に詳細を示す。

【0126】

現在のTUの左上座標を(x0,y0)、現在のスキャン位置を(xC,yC)、TUの幅と高さのlog2の値をそれぞれlog2TbWidth、log2TbHeightとする。また係数の絶対値を格納する配列を、位置(x,y)を用いてAbsLevel[x][y]で表す。図14(a)はabs_remainder、

10

20

30

40

50

または、dec_abs_levelを復号する際に用いる、係数の周りの復号済係数の位置を示している。周りの係数の絶対値和を表す変数をlocSumAbsとすると、TU復号部3024はRRCにおいてlocSumAbsを以下のように導出する。

```

【0127】
locSumAbs = 0
if(xC < (1 < <log2TbWidth)-1) {
locSumAbs += AbsLevel[xC+1][yC]
if(xC < (1 < <log2TbWidth)-2)
locSumAbs += AbsLevel[xC+2][yC]
if(yC < (1 < <log2TbHeight)-1)
locSumAbs += AbsLevel[xC+1][yC+1]
}
if(yC < (1 < <log2TbHeight)-1) {
locSumAbs += AbsLevel[xC][yC+1]
if(yC < (1 < <log2TbHeight)-2)
locSumAbs += AbsLevel[xC][yC+2]
}
locSumAbs0 = locSumAbs
locSumAbs = Clip3(0, 31, locSumAbs-baseLevel*5)
1) 係数位置(xC+1,yC)がブロック内にある (xC < (1 < <log2TbWidth)-1) とき
、locSumAbsにt1(AbsLevel[xC+1][yC])の値を加える。
2) 1) のとき、さらに位置(xC+2,yC)がブロック内にある (xC < (1 < <log2Tb
Width)-2) とき、locSumAbsにt2(AbsLevel[xC+2][yC])の値を加える。
3) 1) のときさらに位置(xC+1,yC+1)がブロック内にある (yC < (1 < <log2Tb
Heigh)-1) とき、locSumAbsにt3(AbsLevel[xC+1][yC+1])の値を加える。
4) 位置(xC,yC+1)がブロック内にある (yC < (1 < <log2TbHeight)-1) とき、l
ocSumAbsにt4(AbsLevel[xC][yC+1])の値を加える。
5) 4) のときさらに位置(xC,yC+2)がブロック内にある (yC < (1 < <log2TbHe
ight)-2) とき、locSumAbsにt5(AbsLevel[xC][yC+2])の値を加える。
以上の処理から計算されたlocSumAbsを変数baseLevelとクリップ処理によって以下
のように更新する。

```

```

【0128】
locSumAbs = Clip3(0, 31, locSumAbs-baseLevel*5)
ここで、baseLevelはシンタックスの種別 (abs_remainderまたはdec_abs_level)
に関連する変数である。baseLevelは、abs_remainderを復号するとき(tccoeff==0)
は4、dec_abs_levelを復号するとき(tccoeff==1)は0としてもよい。tccoeffは復号対
象がabs_remainderであるか、dec_abs_levelであるかを示す変数である。

```

```

【0129】
図14(b)に示すテーブルを用いて、導出されたlocSumAbsからabs_remainder、dec
_abs_levelの復号に用いられるライスパラメータを導出する。

```

```

【0130】
TU復号部3024は、TSRCにおいて周りの復号済係数を用いず、ライスパラメータcRice
Paramを常に1にセットする。

```

```

【0131】
(量子化変換係数の導出、依存量子化)

```

逆量子化・逆変換部311は、変換係数を逆量子化する。量子化方法として、スカラー量子化と依存量子化の2つの方法を備える。依存量子化の場合、さらにRRC部30241において逆量子化処理の一部を行ってもよい。

```

【0132】
逆量子化・逆変換部311は、量子化パラメータqPと、rectNonTsFlag、量子化マトリ

```

ックス $m[x][y]$ の値に応じて、線形スケール値 $ls[x][y]$ を以下のように導出する。逆量子化・逆変換部311は、依存量子化が有効で、かつ、変換スキップが無効の場合と、それ以外とで $ls[x][y]$ の導出方法を切り替える。

【0133】

```
if (sh_dep_quant_used_flag && !transform_skip_flag)
ls[x][y] = (m[x][y]*levelScale[rectNonTsFlag][(qP+1)%6]) << ((qP+1)/6)
else
ls[x][y] = (m[x][y]*levelScale[rectNonTsFlag][qP%6]) << (qP/6)
rectNonTsFlag = (transform_skip_flag==0 && (((Log2(nTbW)+Log2(nTbH)) & 1)==1)) ? 1 : 0
```

10

```
bdShift1 = transform_skip_flag==1 ? 10 : BitDepth+rectNonTsFlag+((Log2(nTbW)+Log2(nTbH))/2)-5+sh_dep_quant_used_flag
```

なお、後述のように係数の範囲を拡張する場合には以下の式で導出してもよい。

```
bdShift1 = BitDepth + rectNonTsFlag + (((Log2(nTbW)+Log2(nTbH))/2) + 10 - log2TransformRange + sh_dep_quant_used_flag
```

(依存量子化ではない場合)

依存量子化ではない場合、量子化変換係数と量子化パラメータから一意に変換係数を導出する。例えば図11、あるいは、図12では以下で変換係数値 d を導出する。

【0134】

```
TransCoeffLevel[x0][y0][cIdx][xC][yC] = AbsLevel[xC][yC] * (1-2*coeff_sign_flag[n]) <式TC_REGULAR>
```

20

```
dz[xC][yC] = TransCoeffLevel[x0][y0][cIdx][xC][yC]
```

```
dnc[xC][yC] = (dz[xC][yC]*ls[xC][yC]+((1 << bdShift1) >> 1)) >> bdShift1
```

```
d[xC][yC] = Clip3( CoeffMin, CoeffMax, dnc[xC][yC] )
```

ここでAbsLevelは量子化変換係数値、ls,bdShift1は量子化パラメータ qP から導出されるパラメータである。CoeffMin、CoeffMaxはクリッピングの最小値と最大値であり、以下の式により導出する。

【0135】

```
CoeffMin = -(1 << log2TransformRange)
```

```
CoeffMax = (1 << log2TransformRange) - 1
```

30

log2TransformRangeは変換係数のレンジを示す。log2TransformRange=15の場合、CoeffMin = -(1 << 15)、CoeffMax = (1 << 15) - 1。

【0136】

変換係数のレンジlog2TransformRangeは、BitDepthに依存して下記により設定してもよい。

【0137】

```
log2TransformRange = extended_precision_processing_flag ? Max(15, BitDepth + BDOFFSET) : 15 (式R-1)
```

ここで、BDOFFSETは変換係数の範囲を計算するための固定値であり、4, 5, 6などが適当である。extended_precision_processing_flagは、変換係数の範囲として拡張レンジを使うか否かを示すフラグである。

40

【0138】

(依存量子化の場合)

一方、図11に示すように、依存量子化は異なるレベルを持つ2つの量子化器を備える。量子化変換係数の中間値(AbsLevelPass1、AbsLevel)のパリティを用いて、4つの状態QStateを切り替える。そして、QStateに応じて量子化・逆量子化する。なお、量子化パラメータを用いた量子化・逆量子化(スケーリング)は別途行われる。

【0139】

```
QState = QStateTransTable[QState][AbsLevelPass1[xC][yC] & 1]
```

```
TransCoeffLevel[x0][y0][cIdx][xC][yC] = (2*AbsLevel[xC][yC]-(QState>1
```

50

? 1 : 0)) * (1-2*coeff_sign_flag[n]) <式TC_DQ>

dz[xC][yC] = TransCoeffLevel[x0][y0][clidx][xC][yC]

dnc[xC][yC] = (dz[xC][yC]*ls[xC][yC]+((1 << bdShift1)>> 1))>> bdShift1

d[xC][yC] = Clip3(CoeffMin, CoeffMax, dnc[xC][yC])

ここで、QStateは状態、QStateTransTable[][]は状態遷移に用いるテーブルで、例えばQStateTransTable[][]={{0,2},{2,0},{1,3},{3,1}}である。

【0140】

QStateは、QStateTransTable[][]を用いずに以下の式で導出してもよい。

【0141】

QState = (32040 >> ((QState << 2) + ((AbsLevelPass1[xC][yC] & 1) << 1))) & 3

10

QStateの値により、AbsLevelが同じでも異なるTransCoeffLevel(あるいはd)が導出される。QStateは1つ前に復号した量子化変換係数値を参照して導出されるので、一般的なスカラー(逆)量子化に比べ、係数間の相関を利用した符号化効率の良い(逆)量子化ができる。

【0142】

(逆変換)

逆コア変換部31123は得られた残差d[][]に対して垂直変換を行う。

【0143】

$e[x][y] = \sum (\text{transMatrix}[y][j] \times d[x][j]) \quad (j = 0..nTbS - 1)$

20

逆コア変換部31123は、第1の中間値e[x][y]をシフト、クリッピングし、第2の中間値g[x][y]を導出する。

【0144】

$g[x][y] = \text{Clip3}(\text{CoeffMin}, \text{CoeffMax}, (e[x][y] + 64) >> 7)$

またCoeffMin、CoeffMaxはクリッピングの最小値と最大値である。

【0145】

逆コア変換部31123は、変換係数d[][],又は、修正変換係数d[][]に対し、変換を行い予測誤差r[][]を導出する(S5)。

【0146】

$r[x][y] = \sum (\text{TransMatrix}[x][j] \times g[j][y]) \quad (j = 0..nTbS - 1)$

30

そして、r[][]に対し、ビットデプス(BitDepth)に応じたシフトを行い、予測画像生成部308で導出される予測画像と同じ精度の誤差resSamples[][]を導出する。例えば、シフトは以下で表現される。

【0147】

$\text{resSamples}[x][y] = (r[x][y] + (1 << (\text{bdShift2} - 1))) >> \text{bdShift2}$ (式BD-1)

ここで、bdShift2 = 5 + log2TransformRange - BitDepthで導出する。(式R-1)

の場合、

$\text{bdShift2} = \text{extended_precision_processing_flag} ? (5 + \text{BDOFFSET}) : \text{Max}(20 - \text{BitDepth}, 0)$

40

通常、予測誤差符号化方法(RRC)では、非ゼロ変換係数が低周波数成分の一部の領域に集中する場合に適するように、LAST位置と呼ばれる非ゼロ変換係数の復号開始位置を符号化し、LAST位置より低周波数成分側をスキャンして量子化変換係数を復号する。また、サインハイディングや依存量子化を使用するため、RRC部30241はこれらの処理に必要な変換係数の絶対値和や、依存量子化の状態(QState)などのパラメータを導出する。一方、変換スキップモードを用いる予測誤差符号化(TSRC)では、非ゼロ変換係数が一部の領域に集中しないため、LAST位置を符号化せず、ブロック全体をスキャンして量子化変換係数を復号する。また、サインハイディングや依存量子化は使用せず、TSRC部30242はこれらの処理に必要な上記パラメータを導出しない。

【0148】

ところで、変換スキップモードの場合でも、通常、予測誤差の符号化方法を用いる方が符

50

号化効率が良い場合がある。そのような場合のために、通常の予測誤差を変換スキップモードの場合に利用してもよい。例えば、図9に示すsh_ts_residual_coding_disabled_flagを通知してもよい。

【0149】

sh_ts_residual_coding_disabled_flagは、現スライスにおいて変換スキップが適用されるブロック（変換スキップブロック）の予測誤差の復号に、TSRCモードであるresidual_ts_coding()が使われるか否かを示す。sh_ts_residual_coding_disabled_flag=1は、変換スキップブロックの予測誤差のパスにRRCモードであるresidual_coding()が使用されることを示す。sh_ts_residual_coding_disabled_flag=0は、変換スキップブロックの予測誤差のパスにresidual_ts_coding()が使用されることを示す。sh_ts_residual_coding_disabled_flagが通知されない場合、0と推定される。

10

【0150】

図10(b)に示すように、!transform_skip_flag[x0][y0][0] || sh_ts_residual_coding_disabled_flagの場合、RRC部30241で処理してもよい。それ以外の場合、TSRC部30242で処理してもよい。つまり、transform_skip_flagが0の場合に加え、sh_ts_residual_coding_disabled_flagが0でない場合にもRRCで変換係数(予測誤差)を通知してもよい。

【0151】

sh_ts_residual_coding_disabled_flagを参照することにより、変換スキップモードの場合でも、通常の予測誤差の符号化方法(RRCモード)を用いることができる。

20

（実施形態2）

係数の絶対値が大きい場合、ライスパラメータの値が小さいと符号量が大きくなり、符号化効率が低下する場合がある。特に、プロフェッショナル向けの映像は、量子化幅が小さいので係数の絶対値が大きくなりやすい。このような場合、図14(b)に示す範囲のライスパラメータの値（0～3）では不適切である。本実施形態では、対象変換係数の周囲の変換係数の絶対値を用いてライスパラメータを導出する方法において、線形予測を用いて、適切なライスパラメータを導出する。

【0152】

（係数の絶対値の予測）

30

TU復号部3024は、実施形態1と同様の処理で、周りの変換係数の絶対値t1～t5の和locSumAbsを導出する。重みa、補正項b、シフト値cを以下の一次式に用い、復号するシンタックス要素（例えば、abs_remainderやdec_abs_level）の値の予測値valを導出する。aは周囲の変換係数の大きさから導出される値locSumAbsに係る重みである。なお、locSumAbsとして、baseLevel*5を引く前の値(locSumAbs0)を用いてもよい（以下同様）。baseLevelはabs_remainderやdec_abs_levelに応じて算出される変数である。

【0153】

val = (a * locSumAbs + b) >> c

なお、valを適当な範囲にクリップしてもよい。

40

【0154】

val = Clip3(0, VMAX, (a * locSumAbs + b) >> c)

ここでVMAXは、valの最大値を示す定数である。

a、b、cを総称して線形予測のパラメータと称してもよい。以下では、条件に応じて、ライスパラメータの導出に用いる線形予測のパラメータを切り替える例を説明する。

【0155】

（シンタックス要素による線形予測のパラメータの場合分け）

シンタックス要素に基づいて線形予測のパラメータa,b,cを切り替える例を説明する。例えば、TU復号部3024は、残差の絶対値のシンタックスに応じて、予測方法を変更してもよい。つまり、残差絶対値のオフセットに対する差分を表すabs_remainderを復号す

50

るときと、残差絶対値dec_abs_levelを復号するときで異なる値のパラメータを用いてもよい。例えば変数tcoeffを用いて、abs_remainderを復号するときをtcoeff=0、dec_abs_levelを復号するときをtcoeff=1で表すと、以下のようにa、b、cを設定してもよい。

```
if (tcoeff == 0) (式LMC-1)
{ a = 3, b = 0, c = 4 } // abs_remainder
else
{ a = 7, b = 16, c = 4 } // dec_abs_level
```

また、cはtcoeffの値に依存せず（例えば常に4）、a、bを以下としてもよい。

```
if (tcoeff == 0)
{ a = 3, b = 0 }
else
{ a = 7, b = 16 }
```

また、TU復号部3024は、シンタックス要素がabs_remainderであるか、dec_abs_levelであるかの判定を、tcoeffではなく、baseLevelにより判定してもよい。つまり、TU復号部3024は、baseLevel==4（もしくはbaseLevel!=0）の場合、シンタックス要素はabs_remainderであり、それ以外（baseLevel==0）の場合、シンタックス要素はdec_abs_levelであるとして判定する。そして、abs_remainderに対する処理、もしくはdec_abs_levelの処理を行ってもよい（以下同様）。

【0156】

（シンタックス要素とブロックサイズによる線形予測のパラメータの場合分け）

TU復号部3024は、シンタックス要素に加え、ブロックサイズ（log2TbWidth、log2TbHeight）に応じて、線形予測のパラメータa、b、cを切り替えてもよい。例えば、以下のように周囲の変換係数の値に係る重みa、補正項b、シフト値cを設定してもよい。

```
if (tcoeff == 0) // abs_remainder, baseLevel != 0
if (log2TbWidth + log2TbHeight < 5) { a = 4, b = 24, c = 4 }
else if (log2TbWidth + log2TbHeight < 6) { a = 4, b = 0, c = 4 }
else if (log2TbWidth + log2TbHeight < 7) { a = 4, b = -24, c = 4 }
else if (log2TbWidth + log2TbHeight < 8) { a = 2, b = 16, c = 4 }
else if (log2TbWidth + log2TbHeight < 9) { a = 2, b = 16, c = 4 }
else { a = 2, b = 16, c = 4 }
else
if (log2TbWidth + log2TbHeight < 5) { a = 8, b = 8, c = 4 }
else if (log2TbWidth + log2TbHeight < 6) { a = 7, b = 16, c = 4 }
else if (log2TbWidth + log2TbHeight < 7) { a = 7, b = 16, c = 4 }
else if (log2TbWidth + log2TbHeight < 8) { a = 6, b = 24, c = 4 }
else if (log2TbWidth + log2TbHeight < 9) { a = 5, b = 32, c = 4 }
else { a = 5, b = 24, c = 4 }
```

また、TU復号部3024はシフト値cを固定し（例えばc=4）、重みa、補正項bをTUサイズに応じて導出してもよい。

```
if (tcoeff == 0) // abs_remainder, baseLevel != 0
if (log2TbWidth + log2TbHeight < 5) { a = 4, b = 24 }
else if (log2TbWidth + log2TbHeight < 6) { a = 4, b = 0 }
else if (log2TbWidth + log2TbHeight < 7) { a = 4, b = -24 }
else if (log2TbWidth + log2TbHeight < 8) { a = 2, b = 16 }
else if (log2TbWidth + log2TbHeight < 9) { a = 2, b = 16 }
else { a = 2, b = 16 }
else
if (log2TbWidth + log2TbHeight < 5) { a = 8, b = 8 }
else if (log2TbWidth + log2TbHeight < 6) { a = 7, b = 16 }
```

```

else if (log2TbWidth + log2TbHeight < 7) { a = 7, b = 16 }
else if (log2TbWidth + log2TbHeight < 8) { a = 6, b = 24 }
else if (log2TbWidth + log2TbHeight < 9) { a = 5, b = 32 }
else { a = 5, b = 24 }

```

上記では、TUサイズが大きくなるほど、重みaを小さく設定することが特徴である。またTU復号部3024は、少なくともtcoeff==1の場合（baseLevel==0の場合）に、TUサイズが大きくなるほど、補正項bを大きくしてもよい。また、条件分岐を用いず、下式によってブロックインデックスbldxを導出してもよい。bldxはブロックサイズとtcoeffから導出されるインデックスである。

【0157】

```
bldx = tcoeff * 6 + Clip3(0, 5, log2TbWidth + log2TbHeight - 5)
```

そして、TU復号部3024はbldxを用いて、図15(a)に示すテーブルからa、b、cを導出してもよい。

【0158】

（シンタックス要素とブロックサイズによる線形予測のパラメータの場合分け（別パターン））

ブロックサイズによる分岐は上記に限定されず、TU復号部3024は、異なるa、b、cの値を用いてもよい。aは周囲の変換係数の値に係る重み、bは補正項b、cはシフト値である。

```

if (tcoeff == 0) // abs_reminder, baseLevel != 0
if (log2TbWidth + log2TbHeight < 5) { a = 4, b = 24, c = 4 }
else if (log2TbWidth + log2TbHeight < 7) { a = 3, b = 8, c = 4 }
else if (log2TbWidth + log2TbHeight < 9) { a = 2, b = 16, c = 4 }
else { a = 2, b = 16, c = 4 }
else
if (log2TbWidth + log2TbHeight < 5) { a = 8, b = 8, c = 4 }
else if (log2TbWidth + log2TbHeight < 7) { a = 7, b = 16, c = 4 }
else if (log2TbWidth + log2TbHeight < 9) { a = 6, b = 24, c = 4 }
else { a = 5, b = 24, c = 4 }

```

また以下でもよい。

```

c = 4
if (tcoeff == 0) // abs_reminder, baseLevel != 0
if (log2TbWidth + log2TbHeight < 5) { a = 4, b = 24 }
else if (log2TbWidth + log2TbHeight < 7) { a = 3, b = 8 }
else if (log2TbWidth + log2TbHeight < 9) { a = 2, b = 16 }
else { a = 2, b = 16 }
else
if (log2TbWidth + log2TbHeight < 5) { a = 8, b = 8 }
else if (log2TbWidth + log2TbHeight < 7) { a = 7, b = 16 }
else if (log2TbWidth + log2TbHeight < 9) { a = 6, b = 24 }
else { a = 5, b = 24 }

```

上記では、TUサイズが大きくなるほど、重みaを小さく設定することが特徴である。またTU復号部3024は、少なくともtcoeff==1の場合（baseLevel==0の場合）、TUサイズが大きくなるほど、補正項bを大きくしてもよい。

【0159】

上記と同様に、条件分岐を用いず、下式によってブロックインデックスbldxを導出してもよい。

【0160】

```
bldx = tcoeff * 4 + Clip3(0, 3, (log2TbWidth + log2TbHeight - 3) / 2)
```

そして、TU復号部3024はbldxを用いて、図15(b)に示すテーブルからa、b、cを導出

してもよい。

【0161】

(シンタックス要素と変換係数位置による線形予測のパラメータの場合分け)

また、TU復号部3024は、シンタックス要素に加え、現在のスキャン位置(xC,yC)に応じて線形予測のパラメータを切り替えてもよい。例えば、下式のように周囲の変換係数の値に係る重みa、補正項b、シフト値cを設定してもよい。

```
if (tcoeff == 0) // abs_reminder, baseLevel != 0
if      (xC + yC < 3) { a = 4, b = 0, c = 4 }
else if (xC + yC < 6) { a = 3, b = 16, c = 4 }
else if (xC + yC < 15) { a = 3, b = 8, c = 4 }
else
    { a = 2, b = 16, c = 4 }
else
if      (xC + yC < 3) { a = 7, b = 16, c = 4 }
else if (xC + yC < 6) { a = 7, b = 16, c = 4 }
else if (xC + yC < 15) { a = 5, b = 32, c = 4 }
else
    { a = 5, b = 24, c = 4 }
```

また以下でもよい。

```
c = 4
if (tcoeff == 0) // abs_reminder, baseLevel != 0
if      (xC + yC < 3) { a = 4, b = 0 }
else if (xC + yC < 6) { a = 3, b = 16 }
else if (xC + yC < 15) { a = 3, b = 8 }
else
    { a = 2, b = 16 }
else
if      (xC + yC < 3) { a = 7, b = 16 }
else if (xC + yC < 6) { a = 7, b = 16 }
else if (xC + yC < 15) { a = 5, b = 32 }
else
    { a = 5, b = 24 }
```

TU復号部3024は、スキャン位置がより高周波数側にシフトする、つまり、xC + yCの和が大きくなるにつれ、重みaをより小さく設定してもよい。

【0162】

(シンタックス要素とサブブロック位置による線形予測のパラメータの場合分け)

また、TU復号部3024は、シンタックス要素に加え、画素単位のスキャン位置ではなく現在のサブブロック位置subSetIdに応じて、線形予測のパラメータを切り替えてもよい。サブブロックは、例えば対象ブロックを4分割した各々の領域であり、subSetIdはブロック内の各サブブロックの位置を示す。例えば、左上を0、右上を1、左下を2、右下を3とする。TU復号部3024は、例えば、下式のように周囲の変換係数の値に係る重みa、補正項b、シフト値cを設定してもよい。

```
if (tcoeff == 0) // abs_reminder, baseLevel != 0
if      (subSetID == 0) { a = 4, b = 8, c = 4 }
else if (subSetID == 1) { a = 4, b = 0, c = 4 }
else if (subSetID == 2) { a = 4, b = -24, c = 4 }
else
    { a = 2, b = 16, c = 4 }
else
if      (subSetID == 0) { a = 7, b = 16, c = 4 }
else if (subSetID == 1) { a = 7, b = 16, c = 4 }
else if (subSetID == 2) { a = 5, b = 32, c = 4 }
else
    { a = 5, b = 24, c = 4 }
```

また以下でもよい。

```
c = 4
```

```

if (tcoeff == 0) // abs_reminder, baseLevel != 0
if      (subSetID == 0) { a = 4, b = 8 }
else if (subSetID == 1) { a = 4, b = 0 }
else if (subSetID == 2) { a = 4, b = -24 }
else      { a = 2, b = 16 }
else
if      (subSetID == 0) { a = 7, b = 16 }
else if (subSetID == 1) { a = 7, b = 16 }
else if (subSetID == 2) { a = 5, b = 32 }
else      { a = 5, b = 24 }

```

10

TU復号部3024は、サブブロック位置がより高周波数側にシフトするにつれ、重みaをより小さく設定してもよい。

【0 1 6 3】

(シンタックス要素と色コンポーネントによる線形予測のパラメータの場合分け)

また、TU復号部3024は、tcoeffに加え、現在の色コンポーネント(cldx)に応じて線形予測のパラメータを切り替えてもよい。例えば、下式のように周囲の変換係数の値に係る重みa、補正項b、シフト値cを設定してもよい。

```

if (tcoeff == 0) // abs_reminder, baseLevel != 0
if (cldx == 0) { a = 3, b = 8, c = 4 }
else      { a = 2, b = 8, c = 4 }
else
if (cldx == 0) { a = 7, b = 24, c = 4 }
else      { a = 6, b = 24, c = 4 }

```

20

(シンタックス要素と色コンポーネントとブロックサイズによる線形予測のパラメータの場合分け)

さらに、TU復号部3024は、tcoeff、現在の色コンポーネント(cldx)に加え、ブロックサイズに応じて線形予測のパラメータを切り替えてもよい。例えば、下式のように重みa、補正項b、シフト値cを設定してもよい。

```

if (tcoeff == 0) // abs_reminder, baseLevel != 0
if (cldx == 0)
if      (log2TbWidth + log2TbHeight < 5) { a = 5, b = 16, c = 4 }
else if (log2TbWidth + log2TbHeight < 7) { a = 3, b = 16, c = 4 }
else if (log2TbWidth + log2TbHeight < 9) { a = 3, b = 0, c = 4 }
else      { a = 2, b = 16, c = 4 }
else
if      (log2TbWidth + log2TbHeight < 5) { a = 5, b = 0, c = 4 }
else if (log2TbWidth + log2TbHeight < 7) { a = 3, b = 16, c = 4 }
else if (log2TbWidth + log2TbHeight < 9) { a = 3, b = 0, c = 4 }
else      { a = 3, b = 16, c = 4 }
else
if (cldx == 0)
if      (log2TbWidth + log2TbHeight < 5) { a = 9, b = 24, c = 4 }
else if (log2TbWidth + log2TbHeight < 7) { a = 7, b = 16, c = 4 }
else if (log2TbWidth + log2TbHeight < 9) { a = 7, b = 16, c = 4 }
else      { a = 5, b = 24, c = 4 }
else
if      (log2TbWidth + log2TbHeight < 5) { a = 8, b = 16, c = 4 }
else if (log2TbWidth + log2TbHeight < 7) { a = 7, b = 24, c = 4 }
else if (log2TbWidth + log2TbHeight < 9) { a = 7, b = 16, c = 4 }
else      { a = 5, b = 24, c = 4 }

```

30

40

50

また以下でもよい。

```

c = 4
if (tcoeff == 0) // abs_reminder, baseLevel != 0
if (cldx == 0)
if (log2TbWidth + log2TbHeight < 5) { a = 5, b = 16}
else if (log2TbWidth + log2TbHeight < 7) { a = 3, b = 16}
else if (log2TbWidth + log2TbHeight < 9) { a = 3, b = 0}
else
{ a = 2, b = 16}
else
if (log2TbWidth + log2TbHeight < 5) { a = 5, b = 0}
else if (log2TbWidth + log2TbHeight < 7) { a = 3, b = 16}
else if (log2TbWidth + log2TbHeight < 9) { a = 3, b = 0}
else
{ a = 3, b = 16}
else
if (cldx == 0)
if (log2TbWidth + log2TbHeight < 5) { a = 9, b = 24}
else if (log2TbWidth + log2TbHeight < 7) { a = 7, b = 16}
else if (log2TbWidth + log2TbHeight < 9) { a = 7, b = 16}
else
{ a = 5, b = 24}
else
if (log2TbWidth + log2TbHeight < 5) { a = 8, b = 16}
else if (log2TbWidth + log2TbHeight < 7) { a = 7, b = 24}
else if (log2TbWidth + log2TbHeight < 9) { a = 7, b = 16}
else
{ a = 5, b = 24}

```

TU復号部3024は、ブロックサイズがより大きくなるにつれ、重みaをより小さく設定してもよい。

【0164】

(シンタックス要素とイントラ/インターによる線形予測のパラメータの場合分け)

TU復号部3024は、パラメータ (tcoeff、baseLevel) に加え、現在のブロックの予測モードCuPredModeに応じて、線形予測のパラメータを切り替えてもよい。線形予測のパラメータはa、b、cや後述のparA、parB、parD、parEなどである。例えば、CuPredModeが、MODE_INTRA、MODE_INTER、MODE_IBCであるかに応じて切り替えてもよいし、MODE_INTRAであるか否かで切り替えてもよい。ブロックの予測モードにより、変換係数の相関に異なる傾向が現れやすい。そこで、予測モードでブロック毎にパラメータを切り替えることにより、予測値が本来の値 (復号する係数の絶対値) からより外れにくくなる。この結果、符号化効率が向上する効果を奏する。

【0165】

(係数の絶対値の予測の別の例)

TU復号部3024は、上記の一次式とは別の式を用いて復号するシンタックス要素の値の予測値valを求めてもよい。TU復号部3024は、実施形態1と同様の処理で、周りの変換係数の絶対値t1～t5の和locSumAbsを導出する。ここでは、locSumAbsに係る重みa、補正項b、シフト値cに、重みdを用いる。dはサブブロックに応じて変化する状態変数remstat (後述) に係る重みである。そして、locSumAbsとremstatに応じて、以下の一次式で予測値valを導出する。

【0166】

```
val = (a * locSumAbs + b + d * remstat) >> c
```

なお、valを適当な範囲にクリップしてもよい。

【0167】

```
val = Clip3(0, VMAX, (a * locSumAbs + b + d * remstat) >> c)
```

ここでVMAXは、valの最大値を示す定数である。

a、b、c、dを総称して線形予測のパラメータと称してもよい。remstatを加えることで、予測値が本来の値（復号する係数の絶対値）からより外れにくくなり、符号化効率が向上する効果を奏する。以下では、remstatについて説明し、条件に応じて、ライスパラメータの導出に用いる線形予測のパラメータを切り替える例を説明する。また状態変数はTU単位で導出してもよい。

【0168】

（状態変数の導出）

TU復号部3024は、復号されたブロック毎に係数の絶対値の平均値を状態変数remstatとしてあらかじめ測定する。そして、remstatをさらに用いて次のサブブロックの予測値valやcRiceParamの値を導出してもよい。remstatは、サブブロック毎に最初に復号した係数の絶対値の大きさを表す変数であってもよい。また、remstatは各係数の大きさを表す変数でもよい

10

（状態変数の導出例1）

例えば、TU復号部3024は、復号した変換係数の値が特定の値TH1を越えていたらremstatをカウントアップする。逆に特定の値TH2を下回ったらremstatをカウントダウンしてもよい。例えば以下のように設定する。

【0169】

TH1 = 3 << (remstat / 4)

TH2 = (1 << (remstat / 4)) / 2

そして、係数の絶対値AbsLevelを用いて、以下のようにremstatを更新してもよい。remstatはスライス、あるいは、タイルごとに初期化し、初期値は0に設定する。また、AbsLevelの代わりに符号化データから復号・符号化したシンタックス値を用いてもよい。例えば、AbsLevelをabs_remainderもしくはdec_abs_levelに置き換えて導出してもよい。

20

【0170】

if (AbsLevel >= TH1)

remstat++

else if ((AbsLevel < TH2) && (remstat > 0))

remstat--

++、--は値のインクリメント、デクリメントを表わす。つまりremstat++はremstat = remstat+1、remstat--はremstat = remstat-1を表す。

30

【0171】

TU復号部3024は、ブロックサイズ、色コンポーネント、変換を用いるか否か（transform_skip_flag）によって、remstatに異なる値を設定してもよい。

【0172】

TH1 = 3 << (remstat[sbType] / 4)

TH2 = 1 << (remstat[sbType] / 4) / 2

if (AbsLevel >= TH1)

remstat[sbType]++

else if ((AbsLevel < TH2) && (remstat[sbType] > 0))

remstat[sbType]--

sbTypeはブロックサイズ、色コンポーネント、transform_skip_flagを用いて導出される変数である。例えば、色コンポーネントとtransform_skip_flagを用いてsbTypeを導出する場合、transform_skip_flag == 0であれば、下式としてもよい。

40

【0173】

sbType = 2 * (cldx == 0) ? 1 : 0

それ以外の場合、下式としてもよい。

【0174】

sbType = (2 * (cldx == 0) ? 1 : 0) + 1

あるいは、下式としてもよい。

50

【0175】

$sbType = (2 * (cldx == 0) ? 1 : 0) + transform_skip_flag$

このとき、sbTypeはブロックサイズに依存しない。

【0176】

また、TU復号部3024は、下式のようにtransform_skip_flagのみを用いてsbTypeを導出してもよい。

【0177】

$sbType = transform_skip_flag$

また、ブロックサイズが4種類、色コンポーネントが2種類、transform_skip_flag=0..1のとき、sbTypeはカウンタを16個所持する。このとき、ブロックサイズを示す変数bgrpを以下のように導出する。

$bgrp = (\log2BlockWidth + \log2BlockHeight < 5) ? 0 :$

$((\log2BlockWidth + \log2BlockHeight < 7) ? 1 :$

$((\log2BlockWidth + \log2BlockHeight < 9) ? 2 : 3))$

ここでtransform_skip_flag == 0の場合、下式としてもよい。

【0178】

$sbType = bgrp * 4 + 2 * (cldx == 0) ? 1 : 0)$

それ以外の場合、下式としてもよい。

【0179】

$sbType = bgrp * 4 + 2 * (cldx == 0) ? 1 : 0) + 1$

あるいは、下式としてもよい。

【0180】

$sbType = bgrp * 4 + 2 * (cldx == 0) ? 1 : 0) + transform_skip_flag$

(状態変数の導出例2)

また、TU復号部3024は、更新率WSによりremstatを更新してもよい。

【0181】

$remstat = (WS * AbsLevel + (((1 < (WSHIFT - WS) * remstat) + (1 < (WSHIFT - 1))) > WSHIFT$

復号対象の変換係数の絶対値(AbsLevel)と既に復号した変換係数に関する変数(remstat)との重み付け和により、remstatを更新する。WSは重み係数であり、WSHIFTは漸近式を整数処理するための値である。例えば、WS = 1、WSHIFT = 2の場合、以下の漸近式である。

【0182】

$remstat = (AbsLevel + (3 * remstat) + 2) > 2$

上式を小数演算で表現すると以下である。

【0183】

$remstat = 0.25 * AbsLevel + 0.75 * remstat$

TU復号部3024は、ブロックサイズ、色コンポーネント、変換を用いるか否か(transform_skip_flag)によって、remstatに異なる値を設定してもよい。sbTypeは(状態変数の導出例1)と同様の方法で算出する。例えば、WS = 1、WSHIFT = 2の場合、下式としてもよい。

【0184】

$remstat[sbType] = (AbsLevel + (3 * remstat[sbType]) + 2) > 2$

例えば、WSHIFT = 3の場合、下式としてもよい。WS=1..7。

【0185】

$remstat[sbType] = (WS * AbsLevel + (8 - WS) * remstat[sbType]) + 4) > 3$

また、上記AbsLevelを、残差を構成するシンタックス値abs_remainderもしくはdec_abs_levelに置き換えて導出してもよい。

【0186】

AbsLevelのみ使ってremstatを導出する場合に比べ、1サブブロック前のremstatと

10

20

30

40

50

現在のサブブロックのAbsLevelの重み演算によりremstatを導出する場合、remstatはAbsLevelの急激な変化（ノイズ）の影響を受けにくくなり、安定するという効果がある。

【0187】

上記のように、TU復号部3024は、各係数の大きさの状態を表す変数として、remstatを各変数単位もしくは各サブブロック単位で導出してもよい。以下、サブブロック単位で導出する場合の具体例をさらに説明する。

【0188】

（サブブロック単位でremstatを導出する例1）

TU復号部3024は、状態変数remstatを、以下の疑似コードに従って導出し、ライスパラメータcRiceParamを導出してもよい。

```
for サブブロック // for(i = lastSubBlock; i >= 0; i--) {
sbTypeを決定する
```

```
for 各係数の復号 // for(n = numSbCoeff - 1; n >= 0; n--) {
cRiceParam = funcRiceParam(remstat[sbType], locSumAbs)
cRiceParamに基づいて残差を復号しAbsLevelを導出する
```

```
if サブブロックの最初の係数の場合 // (n == numSbCoeff - 1)
remstat[sbType] = (AbsLevel + 3 * remstat[sbType]) >> 2
```

ここで、funcRiceParamは、remstatとlocSumAbsからcRiceParamを導出する関数であり、本願に記載の方法のいずれかを利用することができる。サブブロックのループ変数をi、サブブロック内のループ変数をn、サブブロックの最大値をlastSubBlock、サブブロック内の変数の個数をnumSbCoeffとする。numSbCoeff-1を初期値として0に向かってカウントする場合、TU復号部3024は、サブブロックの最初の係数、つまり、n == numSbCoeff - 1の場合にのみ、AbsLevelを用いて、remstatを更新する。

【0189】

なお、サブブロックの最初の係数ではなく、最後の変数(n==0)の場合にremstatを更新する構成であってもよい。上記構成では、remstatはサブブロックに一度のみ更新される。従って、高い精度で残差の大きさを示すremstatから残差のバイナリ長をできるだけ短くするcRiceParamを導出しつつ、remstatの導出にともなうレイテンシを最小限にする効果を奏する。

【0190】

（サブブロック単位でremstatを導出する例2）

TU復号部3024は、状態変数remstatを、以下の疑似コードに従って導出し、ライスパラメータcRiceParamを導出してもよい。

```
for サブブロック // for(i = lastSubBlock; i >= 0; i--) {
sbTypeを決定する
```

```
remstat0 = remstat[sbType]
for 各係数の復号 // for(n = numSbCoeff - 1; n >= 0; n--) {
cRiceParam = funcRiceParam(remstat0, locSumAbs)
cRiceParamに基づいて残差を復号しAbsLevelを導出する
```

```
remstat[sbType] = (AbsLevel + 3 * remstat[sbType]) >> 2
```

なお、上記において、remstat[sbType]の導出をサブブロック毎に一度としてもよい。

【0191】

```
remstat[sbType] = (AbsLevel + 3 * remstat[sbType]) >> 2
```

上式を以下に置き換えた処理としてもよい。

【0192】

if サブブロックの最初の係数の場合

```
remstat[sbType] = (AbsLevel + 3 * remstat[sbType]) >> 2
```

ここで、remstat0は、サブブロックに一度remstat[sbType]を用いて設定される。remstat[sbType]は各変換係数の復号毎に更新されるが、remstatとlocSubAbsの関数

としてcRiceParamを導出する関数の引数にはremstat0を用いる。つまり、残差の復号に用いるcRiceParamの導出には、直前の変換係数の値absLevelに依存するremstatを用いない。従って、直前の変換係数に依存することに伴うレイテンシを低減させることができる。なお、remstat自体の更新は、変数毎に行われるが、直後の残差の復号とは依存関係がなく、並列処理ができるので、レイテンシは発生しない。

【0193】

上記構成では、remstat0はサブブロックに一度のみ更新される。従って、高い精度で残差の大きさを示すremstatから残差のバイナリ長をできるだけ短くするcRiceParamを導出しつつ、remstat0の導出にともなうレイテンシを最小限にする効果を奏する。

【0194】

(サブブロック単位でremstatを導出する例3)

TU復号部3024は、状態変数remstatを、以下の疑似コードに従って導出し、ライスパラメータcRiceParamを導出してよい。

```
for サブブロック // for(i = lastSubBlock; i >= 0; i--) {
sbTypeを決定する
```

```
remstat0 = remstat[sbType]
```

```
for 各係数の復号 // for(n = numSbCoeff - 1; n >= 0; n--) {
```

```
cRiceParam = funcRiceParam(remstat0, locSumAbs)
```

```
cRiceParamに基づいて残差を復号しAbsLevelを導出する
```

```
remstat0 = (AbsLevel + 3 * remstat0) >> 2
```

```
if サブブロックの最初の係数の場合
```

```
remstat[sbType] = (AbsLevel + 3 * remstat[sbType]) >> 2
```

ここで、remstat0は、サブブロックに一度remstat[sbType]を用いて設定される。remstat0は各変換係数の復号毎に更新される。そして、関数funcRiceParamの引数にはremstat0を用いる。remstatは高い精度で残差の大きさを示すので、残差のバイナリ長が短いcRiceParamを導出することができる。一方、remstat[sbType]はサブブロックに一度更新される。これにより、サブブロックを跨いで係数が大きく変化してしまう場合に、導出されたcRiceParamの値が最適なバイナリ長を与えるcRiceParamと大きく異なってしまう現象を抑え、符号化効率が向上する効果を奏する。

【0195】

(各係数単位でremstatを導出する例4)

TU復号部3024は、状態変数remstatを、以下の疑似コードに従って導出し、ライスパラメータcRiceParamを導出してよい。

```
for サブブロック // for(i = lastSubBlock; i >= 0; i--) {
```

```
sbTypeを決定する
```

```
for 各係数の復号 // for(n = numSbCoeff - 1; n >= 0; n--) {
```

```
cRiceParam = funcRiceParam(remstat[sbType], locSumAbs)
```

```
cRiceParamに基づいて残差を復号しAbsLevelを導出する
```

```
remstat[sbType] = (AbsLevel + 3 * remstat[sbType]) >> 2
```

上記構成では、TU復号部3024は、AbsLevelを用いて、sbType毎のremstat[sbType]を各変換係数の復号毎に更新する。そして、関数funcRiceParamの引数にはremstat[sbType]を用いる。remstat[sbType]を変換係数毎に導出するため、残差のバイナリ長が短くなるようなcRiceParamを導出することができる。remstat[sbType]は変換係数の残差の平均的な大きさである。そのため、符号化効率が向上する効果を奏する。

【0196】

(シンタックス要素とブロックサイズによる線形予測のパラメータの場合分け)

TU復号部3024は、tcoeffに応じて線形予測のパラメータを切り替えてもよい。例えば、状態変数の導出例2において周囲の変換係数の値に係る重みa、補正項b、シフト値c、状態変数remstatに係る重みdを以下のように設定してもよい。

```
if (tcoeff == 0) (式LMC-2) // abs_reminder, baseLevel != 0
```

```
{ a = 3, b = 0, c = 4, d = 7 }
else
{ a = 7, b = 16, c = 4, d = 5 }
```

以下のように設定してもよい。

```
if (tcoeff == 0)
{ a = 3, b = 0, c = 4, d = 5 }
else
{ a = 7, b = 16, c = 4, d = 3 }
```

(remstatを用いるRRCの構成の例)

TU復号部3024は、tcoeffに加え、ブロックサイズに応じて、線形予測のパラメータを切り替えてもよい。例えば、状態変数の導出例1において周囲の変換係数の値に係る重みa、補正項b、シフト値c、状態変数remstatに係る重みdを以下のように設定してもよい。

```
if (tcoeff == 0) // abs_reminder, baseLevel != 0
if (log2TbWidth + log2TbHeight < 5) { a = 4, b = 24, c = 4, d = 4 }
else if (log2TbWidth + log2TbHeight < 7) { a = 3, b = 8, c = 4, d = 2 }
else if (log2TbWidth + log2TbHeight < 9) { a = 2, b = 16, c = 4, d = 1 }
else { a = 2, b = 16, c = 4, d = 1 }
```

```
else if (log2TbWidth + log2TbHeight < 5) { a = 8, b = 8, c = 4, d = 3 }
else if (log2TbWidth + log2TbHeight < 7) { a = 7, b = 16, c = 4, d = 2 }
else if (log2TbWidth + log2TbHeight < 9) { a = 6, b = 24, c = 4, d = 0 }
else { a = 5, b = 24, c = 4, d = 1 }
```

また以下でもよい。

```
c = 4
if (tcoeff == 0) // abs_reminder, baseLevel != 0
if (log2TbWidth + log2TbHeight < 5) { a = 4, b = 24, d = 4 }
else if (log2TbWidth + log2TbHeight < 7) { a = 3, b = 8, d = 2 }
else if (log2TbWidth + log2TbHeight < 9) { a = 2, b = 16, d = 1 }
else { a = 2, b = 16, d = 1 }
else if (log2TbWidth + log2TbHeight < 5) { a = 8, b = 8, d = 3 }
else if (log2TbWidth + log2TbHeight < 7) { a = 7, b = 16, d = 2 }
else if (log2TbWidth + log2TbHeight < 9) { a = 6, b = 24, d = 0 }
else { a = 5, b = 24, d = 1 }
```

TU復号部3024は、ブロックサイズがより大きくなるにつれ、重みdをより小さく設定してもよい。

【0197】

また、TU復号部3024は、ブロックサイズが閾値TH3より小さい場合にのみremstatを用いてもよい。

```
if (log2TbWidth + log2TbHeight >= TH3)
val = (a * locSumAbs + b) >> c
else
val = (a * locSumAbs + b + d * remstat) >> c
```

また、状態変数の導出例2において重みa、補正項b、シフト値c、重みdを以下のように設定してもよい。

```
if (tcoeff == 0) // abs_reminder, baseLevel != 0
if (log2TbWidth + log2TbHeight < 5) { a = 4, b = 24, c = 4, d = 6 }
else if (log2TbWidth + log2TbHeight < 7) { a = 3, b = 8, c = 4, d = 5 }
else if (log2TbWidth + log2TbHeight < 9) { a = 2, b = 16, c = 4, d = 3 }
```

```

else { a = 2, b = 16, c = 4, d = 3 }
else
if (log2TbWidth + log2TbHeight < 5) { a = 8, b = 8, c = 4, d = 5 }
else if (log2TbWidth + log2TbHeight < 7) { a = 7, b = 16, c = 4, d = 5 }
else if (log2TbWidth + log2TbHeight < 9) { a = 6, b = 24, c = 4, d = 1 }
else { a = 5, b = 24, c = 4, d = 1 }

```

また以下でもよい。

```
c = 4
```

```

if (tcoeff == 0) // abs_reminder, baseLevel != 0
if (log2TbWidth + log2TbHeight < 5) { a = 4, b = 24, d = 6 }
else if (log2TbWidth + log2TbHeight < 7) { a = 3, b = 8, d = 5 }
else if (log2TbWidth + log2TbHeight < 9) { a = 2, b = 16, d = 3 }
else { a = 2, b = 16, d = 3 }
else
if (log2TbWidth + log2TbHeight < 5) { a = 8, b = 8, d = 5 }
else if (log2TbWidth + log2TbHeight < 7) { a = 7, b = 16, d = 5 }
else if (log2TbWidth + log2TbHeight < 9) { a = 6, b = 24, d = 1 }
else { a = 5, b = 24, d = 1 }

```

TU復号部3024は、ブロックサイズがより大きくなるにつれ、重みdをより小さく設定してもよい。

【0198】

また、TU復号部3024は、ブロックサイズが閾値TH3より小さい場合にのみremstatを用いてもよい。

```
if (log2TbWidth + log2TbHeight >= TH3)
```

```
val = (a * locSumAbs + b) >> c
```

```
else
```

```
val = (a * locSumAbs + b + d * remstat) >> c
```

(シンタックス要素とイントラ/インターによる線形予測のパラメータの場合分け)

さらに、TU復号部3024は、パラメータ (tcoeff、baseLevel) に加え、現在のブロックの予測モードCuPredModeに応じて、線形予測のパラメータを切り替えてもよい。線形予測のパラメータはa、b、c、dであってもよい。例えば、CuPredModeがMODE_INTRA、MODE_INTER、MODE_IBCであるかに応じて切り替えてもよいし、MODE_INTRAであるか否かで切り替えでもよい。ブロックの予測モードにより、変換係数の相関にも異なる傾向が現れやすい。そこで、予測モードでブロックごとのパラメータを切り替えることにより、予測値が本来の値 (復号する係数の絶対値) からより外れにくくなる。この結果、符号化効率が向上する効果を奏する。

【0199】

(TSRCの場合の係数の絶対値の予測)

本実施形態では、TSRCにおいても、対象係数の周囲の係数の絶対値を用いてライスパラメータを導出する方法を用いてもよい。ここで、変換スキップモードの予測誤差符号化方法は、通常の予測誤差符号化方法と比べると係数のスキャン順が逆転する。つまり、TU復号部3024は、TSRCにおいて係数を左上から右下に向かってスキャンしていく。そこで、TU復号部3024は、TSRCにおいて周りの係数の絶対値t1～t5の和locSumAbsを以下のように導出する。図14(c)は、現在のスキャン位置(xC,yC)と周りの係数t1～t5を示す図である。

【0200】

```
locSumAbs = 0
```

```
if( xC > 0 ) {
```

```
locSumAbs += AbsLevel[xC - 1][yC]
```

```
if( xC > 1 )
```

```

locSumAbs += AbsLevel[xC - 2][yC]
if( yC > 1 )
locSumAbs += AbsLevel[xC - 1][yC - 1]
}
if( yC > 0 ) {
locSumAbs += AbsLevel[xC][yC - 1]
if( yC > 1 )
locSumAbs += AbsLevel[xC][yC - 2]
}

```

```
locSumAbs0 = locSumAbs
```

10

```
locSumAbs = Clip3( 0, 31, locSumAbs - baseLevel * 5 )
```

1) 係数位置(xC-1,yC)がブロック内にある (xC > 0) とき、locSumAbsにt1(AbsLevel[xC-1][yC])を加える。

2) 1) のとき、さらに位置(xC-2,yC)がブロック内にある (xC > 1) とき、locSumAbsにt2(AbsLevel[xC-2][yC])を加える。

3) 1) のときさらに位置(xC-1,yC-1)がブロック内にある (yC > 0) とき、locSumAbsにt3(AbsLevel[xC-1][yC-1])を加える。

4) 位置(xC,yC-1)がブロック内にある (yC > 0) とき、locSumAbsにt4(AbsLevel[xC][yC-1])を加える。

5) 4) のときさらに位置(xC,yC-2)がブロック内にある (yC > 1) とき、locSumAbsにt5(AbsLevel[xC][yC-2])を加える。

20

【0201】

TU復号部3024は、上記で求めたlocSumAbs(locSumAbs0)を用いて、locSumAbsに係る重みa、補正項b、シフト値cを以下の一次式に適用し、復号するシンタックス要素（例えば、abs_remainder）の値の予測値valを導出する。

【0202】

```
val = (a * locSumAbs + b) >> c
```

なお、valを適当な範囲にクリップしてもよい。

【0203】

```
val = Clip3(0, VMAX, (a * locSumAbs + b) >> c)
```

30

ここでVMAXは、valの最大値を示す定数である。

a、b、cを総称して線形予測のパラメータと称してもよい。例えば、TU復号部3024は、線形予測のパラメータa、b、cをそれぞれ以下のように設定してもよい。

【0204】

a = 4, b = 64, c = 4 (式LMTSRC-1)

(remstatを用いるTSRCの構成の例)

TU復号部3024は、RRCの場合の係数の絶対値の予測と同様に、remstatを用いて復号するシンタックス要素の値の予測値valを求めてもよい。すなわち、locSumAbsに係る重みa、補正項b、シフト値cに、状態変数remstatに係る重みdを用いる。そして、locSumAbsとremstatに応じて、以下の一次式で予測値valを導出する。

40

【0205】

```
val = (a * locSumAbs + b + d * remstat) >> c
```

なお、valを適当な範囲にクリップしてもよい。

【0206】

```
val = Clip3(0, VMAX, (a * locSumAbs + b + d * remstat) >> c)
```

ここでVMAXは、valの最大値を示す定数である。

a、b、c、dを総称して線形予測のパラメータと称してもよい。remstatを加えることで、予測値が本来の値（復号する係数の絶対値）からより外れにくくなり、符号化効率が向上する効果を奏する。

【0207】

50

例えば、TU復号部3024は、線形予測のパラメータa、b、c、dをそれぞれ以下のように設定してもよい。

【0208】

$a = 4, b = 64, c = 4, d = 1$ (式LMTSRC-2)

TU復号部3024は、重みaを0となるように、線形予測のパラメータa、b、c、dを導出してもよい。例えば、それぞれ以下のように設定してもよい。

【0209】

$a = 0, b = 0, c = 4, d = 22$ (式LMTSRC-3)

あるいは、以下のように設定してもよい。

【0210】

$a = 0, b = 24, c = 4, d = 15$ (式LMTSRC-4)

つまり、TU復号部3024は、remstatから、変換係数の予測値valを以下の式で導出してもよい。

【0211】

$val = (22 * remstat) \gg 4)$

$val = (24 + 15 * remstat) \gg 4)$

さらに簡略化して以下でもよい。

【0212】

$val = (11 * remstat) \gg 3)$

$val = (12 + 8 * remstat) \gg 3)$

さらに簡略化して以下でもよい。

【0213】

$val = (3 * remstat) \gg 1)$

$val = (3 + 2 * remstat) \gg 1)$

変換スキップモードの予測誤差符号化方法においては、当該ブロックの係数と周囲の係数の間に相関がない場合がある。その場合、重みaの重みをなくし($a=0$)、remstatのみを使用する方が符号化効率が向上する場合がある。

【0214】

(locSumAbsの別の導出方法)

対象変換係数の1つ前に復号した係数のlocSumAbsを、変数prevLocSumAbsに保持し、prevLocSumAbsを用いて対象変換係数のlocSumAbsを導出してもよい。locSumAbsは、対象変換係数の周りの複数の変換係数の絶対値t1~t5の和である。例えば、TU復号部3024は、下式のようにt1~t5の和をlocSumAbsに格納した後、locSumAbsを更新してもよい。

【0215】

$locSumAbs = t1 + t2 + t3 + t4 + t5$

$locSumAbs = (locSumAbs + prevLocSumAbs) \gg 1$

対象変換係数の周りの係数の和(LocSumAbs)と、1つ前に復号した変換係数の周りの係数の和(prevLocSumAbs)との平均をとることによって、LocSumAbsの急激な変化が抑制され、ノイズの影響を受けにくくなるという効果を奏する。

【0216】

更新率(重み係数)WLは0.5に限定されず、以下でもよい。

【0217】

$locSumAbs = (WL * locSumAbs + ((1 \ll WLSHIFT) - WL) * prevLocSumAbs + 1 \ll (WLSHIFT - 1)) \gg WLSHIFT$

WLSHIFTは更新率によるlocSumAbsの導出を整数処理するための値である。

【0218】

(ライスパラメータ導出式)

ある値に対してゴロム・ライス符号によるバイナリ化を行う際、バイナリの長さを最小にするライスパラメータを一意に決めることが可能である。TU復号部3024は、上記によ

10

20

30

40

50

り得られた予測値valを基に、以下の式でライスパラメータcRiceParamを導出する。

【0219】

$cRiceParam = val \ll 3 ? 0 : \text{floor}(\log_2(val / 3)) + 1$ (式LMC-3)

2を底とする対数値、またそのfloorによる整数値は、BitScanReverseあるいはclzのような組み込み命令と呼ばれる処理によってCPUによるソフトウェア及びハードウェアで高速に演算可能である。したがって、上記の例によれば、2を底とする対数値を用いることにより、容易にライスパラメータを導出する効果を奏する。

【0220】

また、ハードウェアへの実装を考慮し、割り算をシフト演算に置き換えて、ライスパラメータcRiceParamを導出してもよい。図16は、本実施形態に係る予測値valからcRiceParamを導出する一般式の例である。図16(a)に示すように、傾きparA、オフセットparB、シフト値parCを用いてcRiceParamを導出してもよい。例えば、(式LMC-3)において、以下のようにcRiceParamを導出してもよい。

【0221】

$cRiceParam = \text{floor}(\log_2((val * 5) \gg 4)) + 1$ (式LMC-4)

ここでは、(式LMC-3)のvalの係数である1/3を近似して5/16に置き換えているが、1/4, 3/8, 5/16, 11/32, 21/64に置き換えてもよい、つまり、val/3をval>>2, (val*3)>>3, (val*5)>>4, (val*11)>>5, (val*21)>>6で置き換えて計算してもよい(以下同様)。

また、log2の引数として0を避けるため以下でもよい。

【0222】

$cRiceParam = val * 5 \ll 16 ? 0 : \text{floor}(\log_2((val * 5) \gg 4)) + 1$

また以下でもよい。

【0223】

$cRiceParam = val \ll 3 ? 0 : \text{floor}(\log_2((val * 5) \gg 4)) + 1$

また、図16(a)においてシフト演算をlog2の外に出すことにより、図16(b)に示すように、傾きparA、オフセットparBを用いてcRiceParamを導出してもよい。例えば、(式LMC-3)において、以下のようにcRiceParamを導出してもよい。

【0224】

$cRiceParam = \text{floor}(\log_2(val * 5)) - 3$

また、log2の引数として0を避けるため以下でもよい。

【0225】

$cRiceParam = val == 0 ? 0 : \text{floor}(\log_2(val * 5)) - 3$

また、最小値を0以上、最大値をNC以下にクリップしてもよい。NCは例えば15。

【0226】

$cRiceParam = \text{Clip3}(0, NC, cRiceParam)$

以上の処理により、変換係数の絶対値が大きくなった場合でも、絶対値に応じた適切なライスパラメータを導出することにより、符号化効率が向上する効果を奏する。

【0227】

(導出式の例1)

locSubAbsの線形予測とcRiceParam導出の計算は一連の計算としてマージしてもよい。

【0228】

$val = (a * \text{locSumAbs} + b) \gg c$

$cRiceParam = \text{floor}(\log_2((val * 5) \gg 4)) + 1$

これらの式を一つの式で導出してもよい。たとえば、TU復号部3024は、locSumAbsの積和とシフトと2を底とする対数値及びオフセットの加算から、以下のようにcRiceParamを導出してもよい。

【0229】

$cRiceParam = \text{floor}(\log_2((aK * \text{locSumAbs} + bK) \gg cK)) + 1$

より一般には、図17(a)に示すように傾きparA (=aK), オフセットparB (=bK), シフト値parC (=cK), オフセットparEを用いて、下式でcRiceParamを表してもよい。

【0230】

cRiceParam = floor(log2((parA * locSumAbs + parB) >> parC)) + parE
ここで、log2の引数が1未満になることを避けるため、以下の式からcRiceParamを導出してよい。

【0231】

cRiceParam = locSumAbs < TK ? 0 : floor(log2((parA * locSumAbs + parB) >> parC)) + parE

さらに、log2()の引数が1未満になることを避けるために、log2の中の項で1を加算した以下の式でもよい。

【0232】

cRiceParam = floor(log2(((parA * locSumAbs + parB) >> parC) + 1)) + parE

なお、ここでaK = 5 * a、bK = 5 * b、cK = (c+4)であってもよい。ここで、a、b、cは本願の導出方法をそのまま利用できる。またTKはlog2の引数が1未満になることを避けるための閾値であり、(aK * locSumAbs + bK) >> cK = 1となるlocSumAbsをTKとしてもよい。つまり、TKは下記で導出してよい。

【0233】

TK = ceil(((1 < cK) - bK) / aK)

例えば(式LMC-1)を用いて、TU復号部3024は、以下のようにcRiceParamを導出してよい。

【0234】

```
if (baseLevel != 0)          // a = 3 (aK=15), b = 0 (bK=0), c = 4 (cK=8)
cRiceParam = locSumAbs < 18 ? 0 : floor(log2((15 * locSumAbs + 0) >
> 8)) + 1
else                          // a = 7 (aK=35), b = 16 (bK=80), c = 4 (cK=8)
cRiceParam = locSumAbs < 6 ? 0 : floor(log2((35 * locSumAbs + 80) >
> 8)) + 1
```

ここで baseLevel != 0は、対象とする残差のシンタックス要素がabs_reminderであることを意味し、tcoeff==0で判定してもよい(以下、同様)。

【0235】

また、TSRCの場合も、(式LMTSRC-1)を用いて、TU復号部3024は、以下のようにcRiceParamを導出してよい。

【0236】

cRiceParam = floor(log2((20 * locSumAbs + 320) >> 8)) + 1

式を整理して以下のようにしてもよい。

【0237】

cRiceParam = floor(log2((5 * locSumAbs + 80) >> 6)) + 1

(remstatを用いる場合の導出式の例1)

上記(導出式の例1)は、remstatを用いる場合に適用してもよい。

【0238】

TU復号部3024は、locSumAbsの積和とremstatの積和と、シフトと2を底とする対数値及びオフセットの加算から、以下のようにcRiceParamを導出してよい。

【0239】

cRiceParam = floor(log2((aK * locSumAbs + bK + dK * remstat) >> cK)) + 1

より一般には、図17(d)に示すように傾きparA (=aK), オフセットparB (=bK), 傾きparD (=dK), シフト値parC (=cK), オフセットparEを用いて以下の式からcRiceParamを導出してよい。

【0240】

cRiceParam = floor(log2((parA * locSumAbs + parB + parD * remstat) >

10

20

30

40

50

〉 parC)) + parE

ここで、log2の引数が1未満になることを避けるため、以下の式からcRiceParamを導出してよい。

【0241】

```
cRiceParam = parA * locSumAbs + parD * remstat < (1 < (parC) - parB
? 0 : floor(log2( (parA * locSumAbs + parB + parD * remstat) > > parC )
) + parE
```

なお、ここでaK = 5 * a、bK = 5 * b、cK = (c+4)、dK = 5 * dであってもよい。a、b、c、dの導出には、本願の（remstatを用いるRRCの構成の例）、（remstatを用いるTSRCの構成の例）に示した導出方法をそのまま利用できる。例えば（式LMC-2）の場合には、以下のようにcRiceParamを導出してよい。

【0242】

```
if (baseLevel != 0)          // a = 3 (aK=15), b = 0 (bK=0), c = 4 (cK=8), d
= 7 (dK=35)
cRiceParam = 15 * locSumAbs + 35 * remstat < 256 ? 0 : floor(log2((15
* locSumAbs + 0 + 35 * remstat) > > 8)) + 1
else                          // a = 7 (aK=35), b = 16 (bK=80), c = 4 (cK=8), d = 5 (dK=25)
cRiceParam = 35 * locSumAbs + 25 * remstat < 176 ? 0 : floor(log2((35
* locSumAbs + 80 + 25 * remstat) > > 8)) + 1
```

また、TSRCの場合も、TU復号部3024は（式LMTSRC-2）、（式LMTSRC-3）、（式LMTSRC-4）のいずれかを用いて、以下のようにcRiceParamを導出してよい。ここで、aK=4*5=20、bK = 64*5=320、cK=4+4=8である。

【0243】

```
cRiceParam = floor(log2((20 * locSumAbs + 320 + 5 * remstat) > > 8)) + 1
cRiceParam = remstat < 3 ? 0 : floor(log2(( 110 * remstat) > > 8)) + 1
cRiceParam = remstat < 2 ? 0 : floor(log2((120 + 75 * remstat) > > 8)) + 1
またさらに最小値を0以上、最大値をNC以下にクリップしてもよい。
```

【0244】

```
cRiceParam = Clip3(0, NC, cRiceParam)
```

上記、導出式の例1では、locSumAbsと傾きaK、バイアスbKの線形式とシフトから導出される値の対数値から、cRiceParamを簡易な式で導出できるという効果を奏する。また傾きaKとバイアスbKをシンタックス値や、ブロックサイズ、ブロック内位置に応じて変更することもできる。

【0245】

すでに説明したように2を底とする対数値、またそのfloorによる整数値は、BitScanReverseかclzと呼ばれる組み込み命令と呼ばれる処理によってCPUによるソフトウェア及びハードウェアで高速に演算可能である。したがって、上記の例によれば、2を底とする対数値を用いることにより、容易にライスパラメータを導出する効果を奏する。

【0246】

（導出式の例2）

また、（導出式の例1）において、TU復号部3024は、シフト演算をlog2の外に出すことにより、locSumAbsの積和とシフトと2を底とする対数値及びオフセットの加算から以下の式でcRiceParamを導出してよい。

【0247】

```
cRiceParam = floor(log2(aK * locSumAbs + bK)) - cK2
```

ここで、cK2 = (c+4-1) = c+3としてもよい。

【0248】

また、cRiceParamが0以上であればよいので以下でもよい。TKは（導出式の例1）と同じ値である。

【0249】

10

20

30

40

50

```
cRiceParam = locSumAbs < TK ? 0 : floor(log2(aK * locSumAbs + bK)) - cK2
cRiceParam = Max(0, floor(log2(aK * locSumAbs + bK)) - cK2)
```

例えば（式LMC-1）を用いて、TU復号部3024は、以下のようにcRiceParamを導出してもよい。

【0250】

```
if (baseLevel != 0) // a = 3, b = 0, c = 4
cRiceParam = Max(0, floor(log2(15 * locSumAbs + 0)) - 7)
else // a = 7, b = 16, c = 4
cRiceParam = Max(0, floor(log2(35 * locSumAbs + 80)) - 7)
```

また最小値を0以上にするだけでなく、最大値をNC以下にクリップしてもよい。

【0251】

```
cRiceParam = Clip3(0, NC, floor(log2(35 * locSumAbs + 80)) - 7)
```

さらに簡略化構成として、TU復号部3024は以下のようにcRiceParamを導出してもよい。

【0252】

```
if (baseLevel != 0) // tcoeff == 0
cRiceParam = Max(0, floor(log2(15 * locSumAbs)) - 7)
else
cRiceParam = Max(0, floor(log2(9 * locSumAbs) + 20) - 5)
```

さらに簡略化構成として、TU復号部3024は以下のようにcRiceParamを導出してもよい。

【0253】

```
if (baseLevel != 0) // tcoeff == 0
cRiceParam = Max(0, floor(log2(4 * locSumAbs)) - 5)
else
cRiceParam = Max(0, floor(log2(9 * locSumAbs) + 20) - 5)
```

上式と等価な処理として、TU復号部3024は以下のようにcRiceParamを導出してもよい。

【0254】

```
if (baseLevel != 0) // tcoeff == 0
cRiceParam = Max(0, floor(log2(locSumAbs)) - 3)
else
cRiceParam = Max(0, floor(log2(9 * locSumAbs) + 20) - 5)
```

より一般には、図17(b)に示すように、TU復号部3024は、locSumAbs、傾きparA (=aK)、オフセットparB (=bK)、オフセットparE (=cK2)を用いて以下の式からcRiceParamを導出してもよい。

【0255】

```
cRiceParam = floor(log2((parA * locSumAbs + parB)) + parE)
```

また、TSRCの場合も、TU復号部3024は（式LMTSRC-1）を用いて、cRiceParamを以下のように導出してもよい。

【0256】

```
cRiceParam = Max(0, floor(log2(5 * locSumAbs + 80)) - 5)
```

図17(e)に示すようにTU復号部3024は、locSumAbsとremstatと、傾きparA (=aK)、オフセットparB (=bK)、傾きparD (=dK)、オフセットparE (=cK2)を用いて以下の式からcParamを導出してもよい。

【0257】

```
cRiceParam = floor(log2((parA * locSumAbs + parB + parD * remstat)) + parE)
```

例えば（式LMC-2）を用いて、TU復号部3024は、以下のようにcRiceParamを導出してもよい。

【0258】

10

20

30

40

50

```

if (baseLevel != 0)          // a = 3, b = 0, c = 4, d = 7
cRiceParam = Max(0, floor(log2(15 * locSumAbs + 35 * remstat)) - 7)
else                          // a = 7, b = 16, c = 4, d = 5
cRiceParam = Max(0, floor(log2(35 * locSumAbs + 80 + 25 * remstat)) - 7)

```

また、TU復号部3024は、上記parA、parB、parD、parE以外の値を用い、以下のよう
にcRiceParamを導出してもよい。

【0259】

```

if (baseLevel != 0)          // parA = 1, parB = 1, parD = 13, parE = -5
cRiceParam = Max(0, floor(log2(locSumAbs + 1 + 13 * remstat)) - 5)
else                          // parA = 6, parB = 11, parD = 14, parE = -5
cRiceParam = Max(0, floor(log2(6 * locSumAbs + 11 + 14 * remstat)) - 5)

```

すなわち、remstatを用いる場合、baseLevel!=0(abs_reminder)の場合とbaseLevel==0(dec_abs_level)の場合に同じオフセットの値parEを用いてもよい（以下同様）。また、baseLevel!=0(abs_reminder)の場合のremstatに対する傾きparDの値を、baseLevel==0(dec_abs_level)のparDよりも小さくしてもよい。

【0260】

また、以下のようにcRiceParamを導出してもよい。

【0261】

```

if (baseLevel != 0)          // parA = 1, parB = 0, parD = 12, parE = -5
cRiceParam = Max(0, floor(log2(locSumAbs + 12 * remstat)) - 5)
else                          // parA = 6, parB = 10, parD = 12, parE = -5
cRiceParam = Max(0, floor(log2(6 * locSumAbs + 10 + 12 * remstat)) - 5)

```

また、TU復号部3024は、以下のようにcRiceParamを導出してもよい。

【0262】

```

if (baseLevel != 0)          // parA = 1, parB = 0, parD = 12, parE = -5
cRiceParam = Max(0, floor(log2(locSumAbs + 12 * remstat)) - 5)
else                          // parA = 6, parB = 0, parD = 13, parE = -5
cRiceParam = Max(0, floor(log2(6 * locSumAbs + 13 * remstat)) - 5)

```

さらに、TU復号部3024は、（remstatを用いるRRCの構成の例）における（シンタックス要素とイントラ/インターによる線形予測のパラメータの場合分け）の実施例を用いて、以下のようにcRiceParamを導出してもよい。

【0263】

```

if (baseLevel != 0)
if (CuPredMode == MODE_INTRA)
cRiceParam = Max(0, floor(log2(15 * locSumAbs + 35 * remstat)) - 7)
else // CuPredMode == MODE_INTER, MODE_IBC
cRiceParam = Max(0, floor(log2(locSumAbs + 12 * remstat)) - 5)
else // baseLevel == 0
if (CuPredMode == MODE_INTRA)
cRiceParam = Max(0, floor(log2(35 * locSumAbs + 80 + 25 * remstat)) - 7)
else // CuPredMode == MODE_INTER, MODE_IBC
cRiceParam = Max(0, floor(log2(6 * locSumAbs + 13 * remstat)) - 5)

```

上記に示すように、イントラである場合のlocSumAbsに係る傾きの値をインターである場合の傾きの値よりも大きくしてもよい。同様に、remstatに係るイントラである場合のremstatに係る傾きの値をインターである場合の傾きの値よりも大きくしてもよい。

【0264】

また、TSRCの場合も、（式LMTSRC-2）、（式LMTSRC-3）、（式LMTSRC-4）のいずれかを用いて、TU復号部3024は、以下のようにcRiceParamを導出してもよい。

【0265】

```

cRiceParam = Max(0, floor(log2(20 * locSumAbs + 320 + 5 * remstat)) - 7)

```

```
cRiceParam = Max(0, floor(log2( 110 * remstat) - 7)
cRiceParam = Max(0, floor(log2(120 + 75 * remstat) - 7)
```

上記、(導出式の例2)では、locSumAbsとaKおよびbKの線形式から導出される値との対数値とcK2から、cRiceParamを簡易な式で導出できるという効果を奏する。またaKとbKをシンタックス値、ブロックサイズ、ブロック内位置に応じて変更することもできる。

【0266】

(導出式の例3)

(導出式の例2)において、傾きaKの値を2の累乗として近似することによって、cRiceParamはシフト処理で簡易に計算することができる。つまりTU復号部3024は、locSumAbsのシフトと加算、2を底とする対数値及びオフセットの加算から、以下のようにcRiceParamを導出してよい。

【0267】

```
cRiceParam = Max(0, floor(log2((locSumAbs < <aS) + bK)) - cK2)
```

ここで、aS=floor(log2(aK)+0.5)。Max()は、導出されるcRiceParamの最小値が0以上であることを保証するための演算である。さらに最小値を考慮してクリップしてもよい。

【0268】

```
cRiceParam = Clip3(0, NC, floor(log2((locSumAbs < <aS) + bK)) - cK2)
```

ここで、クリップに用いる最大値NCは、例えば15などでもよい。

【0269】

より具体的には、TU復号部3024は(式LMC-1)の値(15,35)を2の累乗(16,32)に丸めて、以下のようにcRiceParamを導出してよい。

【0270】

```
if (baseLevel != 0) // tcoeff == 0
```

```
cRiceParam = Max(0, floor(log2(locSumAbs < <4)) - 7)
```

```
else
```

```
cRiceParam = Max(0, floor(log2(locSumAbs < <5) + 80) - 7)
```

さらに簡略化して、TU復号部3024は以下のようにcRiceParamを導出してよい。

【0271】

```
if (baseLevel != 0) // tcoeff == 0
```

```
cRiceParam = Max(0, floor(log2(locSumAbs)) - 3)
```

```
else
```

```
cRiceParam = Max(0, floor(log2(locSumAbs) + 5) - 2)
```

より一般には、図17(c)に示すようにシフト値parS (=aS), オフセットparB (=bK), オフセットparE (=cK2)を用いて以下の式からcRiceParamを導出してよい。

【0272】

```
cRiceParam = floor(log2( (locSumAbs < <parS) + parB)) + parE
```

また、TSRCの場合も、TU復号部3024は(式LMTSRC-1)の値を2の累乗に丸めて、locSumAbsとシフト処理、log2処理を用いて、以下のようにcRiceParamを導出してよい。

【0273】

```
cRiceParam = Max(0, floor(log2((locSumAbs < <2) + 80)) - 5)
```

さらに簡略化して、TU復号部3024は以下のようにcRiceParamを導出してよい。

【0274】

```
cRiceParam = Max(0, floor(log2(locSumAbs + 20)) - 3)
```

TU復号部3024は、図17(f)に示すように、locSumAbs、remstat、傾きシフト値parS (=aS), オフセットparB (=bK), シフト値parT (=dT), オフセットparE (=cK2)を用いて以下の式からcRiceParamを導出してよい。

【0275】

10

20

30

40

50

```
cRiceParam = floor(log2( (locSumAbs < <parS) + parB + (remstat < <parT) )) + parE
```

例えば（式LMC-2）を用いて、TU復号部3024は、以下のようにcRiceParamを導出してもよい。

【0276】

```
if (baseLevel != 0) // tcoeff == 0
```

```
cRiceParam = Max(0, floor(log2((locSumAbs < <4) + (remstat < <5)) ) - 7)
else
```

```
cRiceParam = Max(0, floor(log2((locSumAbs < <5) + 80 + (remstat < <5)) ) - 7)
```

10

さらに簡略化して、TU復号部3024は以下のようにcRiceParamを導出してもよい。

【0277】

```
if (baseLevel != 0) // tcoeff == 0
```

```
cRiceParam = Max(0, floor(log2(locSumAbs + (remstat < <1)) ) - 3)
else
```

```
cRiceParam = Max(0, floor(log2(locSumAbs + 1 + remstat)) - 2)
```

また、TSRCの場合も、TU復号部3024は（式LMTSRC-2）、（式LMTSRC-3）、（式LMTSRC-4）のいずれかを用いて、以下のようにcRiceParamを導出してもよい。

【0278】

```
cRiceParam = Max(0, floor(log2(5 * locSumAbs + 80 + remstat) - 5))
```

20

```
cRiceParam = Max(0, floor(log2(remstat)))
```

```
cRiceParam = Max(0, floor(log2(15 + 10 * remstat) - 4))
```

```
cRiceParam = Max(0, floor(log2(4 + 3 * remstat) - 2))
```

上記、導出式の例3では、locSumAbsとシフト値aS、バイアスBKの線形式とシフトから、cRiceParamを簡易な式で導出できるという効果を奏する。

【0279】

（線形予測のパラメータの伝送）

ヘッダ復号部3020は、符号化データからライスパラメータの予測に利用される重みa、補正項b、状態変数remstatに係る重みdを復号してもよい。また、TU復号部3024は、線形予測のパラメータの値を用いて上述の導出方法、例えば導出例1～3のいずれかを用い、ライスパラメータcRiceParamを導出し、残差絶対値差分abs_remainderと残差絶対値dec_abs_levelを復号してもよい。線形予測のパラメータとは、復号した線形予測に係る重みa、補正項b、重みd等である。

30

【0280】

より具体的には、上述のparA, parB, parC, parE, parSもしくはその一部を符号化データから復号して図17の式(a)、式(b)、式(c)などの線形予測により、cRiceParamを導出してもよい。

【0281】

図18(a)は、本実施形態のパラメータのシンタックステーブルの例である。ヘッダ復号部3020は、ライスパラメータを通知するか否かを示すフラグrice_adaptation_enabled_flagをSPSもしくはPPS、スライスヘッダから復号する。rice_adaptation_enabled_flagが1の場合に、slice_rice_paramA_minus1、slice_rice_paramB、slice_paramC、slice_paramDを復号する。例えば、parA = slice_rice_paramA_minus1+1、parB = slice_rice_paramB、parC = slice_rice_paramC、parD = slice_rice_paramDのように導出する。さらに、図17(a)の式により、parA, parB, parC, parDと、locAbsSum (=locAbsSum0)を用いてcRiceParamを導出してもよい。なお、slice_rice_paramA_minus1のみ固定のオフセットを1からの差分値としている。しかし、これに限定されず、所定のオフセットVA, VB, VC, VDを用いたシンタックス要素slice_rice_paramA_minusVA、slice_rice_paramA_minusVB、slice_rice_paramA_minusVC、slice_rice_paramA_minusVDを復号し、parA, par

40

50

B, parC, parDを以下のように導出してもよい。

【0282】

```
parA = slice_rice_paramA_minusVA + VA
parB = slice_rice_paramB_minusVB + VB
parC = slice_rice_paramC_minusVC + VC
parD = slice_rice_paramD_minusVD + VD
```

また、さらに、abs_remainderの復号に用いるパラメータと、dec_abs_levelの復号に用いるパラメータを異なるシンタックスで復号してもよい。例えば、slice_rice_rem_paramA、slice_rice_rem_paramB、slice_rice_rem_paramC、slice_rice_rem_paramDを復号する。そして、abs_remainderの場合にこれらのパラメータを用いてライスパラメータを導出し、abs_remainderを復号してもよい。また、slice_rice_abs_paramA、slice_rice_abs_paramB、slice_rice_abs_paramC、slice_rice_abs_paramDを復号する。そして、dec_abs_levelの場合にこれらのパラメータを用いてライスパラメータを導出して、dec_abs_levelを復号してもよい。図18(b), (c)は、パラメータの種類は異なるが、abs_remainderとdec_abs_levelで異なるシンタックスを用いる例である。

10

【0283】

さらに、TSRCの場合に、さらに異なるシンタックスを復号して、TSRCの残差の復号に用いるライスパラメータの導出に用いてもよい。例えば、slice_rice_ts_paramA、slice_rice_ts_paramB、slice_rice_ts_paramC、slice_rice_ts_paramDの全て、もしくは、一部を復号する。そして、parA, parB, parC, parDを導出し、TSRCの残差abs_remainderを復号してもよい。

20

【0284】

図18(b)は、本実施形態のパラメータのシンタックステーブルの例である。ヘッダ復号部3020は、ライスパラメータを通知するか否かを示すフラグrice_adaptation_enabled_flagをSPSもしくはPPS、スライスヘッダから復号する。rice_adaptation_enabled_flagが1の場合に、slice_rice_xxx_paramA_minus1、slice_rice_xxx_paramB、slice_rice_xxx_paramDを復号する。図に示すようにxxxはrem, absのいずれかを設定する。そして、abs_remainderの復号に用いるパラメータと、dec_abs_levelの復号に用いるパラメータを異なるシンタックス要素を用いて復号してもよい。例えば、parA = slice_rice_xxx_paramA_minus1 + 1、parB = slice_rice_xxx_paramB、parD = slice_rice_xxx_paramDのように導出する。abs_remainderではxxxにはrem、それ以外ではxxxにはabsを設定する。ここでもシンタックス値は所定のオフセットを用いて符号化されていてもよい。この場合、ヘッダ復号部3020は、所定のオフセットVA, VB, VDを各々復号したシンタックスに加算する。さらに、図17(b)の式により、parA, parB, parDと、locAbsSum (=locAbsSum0)を用いてcRiceParamを導出してもよい。

30

【0285】

図18(c)は、本実施形態のパラメータのシンタックステーブルの例である。ヘッダ復号部3020は、ライスパラメータを通知するか否かを示すフラグrice_adaptation_enabled_flagをSPSもしくはPPS、スライスヘッダから復号する。rice_adaptation_enabled_flagが1の場合に、slice_rice_xxx_params_minus1、slice_rice_xxx_paramB、slice_rice_xxx_paramDを復号する。例えば、parS = slice_rice_xxx_params_minus1 + 1、parB = slice_xxx_rice_paramB、parD = slice_xxx_rice_paramDのように導出する。すでに説明したように、abs_remainderとdec_abs_levelで異なるシンタックス要素を用いてもよく、abs_remainderであるか、もしくは、dec_abs_levelであるかに応じてパラメータを切り替えてもよい。また、所定のオフセットVS, VB, VDを、各々復号したシンタックス要素に加算してもよい。さらに、図17(c)の式により、parS, parB, parDと、locAbsSum (=locAbsSum0)を用いてcRiceParamを導出してもよい。

40

50

【0286】

ヘッダ復号部、TU復号部3024は、上記によらず、パラメータをシーケンスごとに伝送してもよい。また、パラメータをピクチャごとに伝送してもよいし、スライスごとに伝送してもよい。このように、シーケンス、ピクチャ、スライスなどの状態に応じた適切なパラメータを伝送することで、予測値が本来の値（復号する係数の絶対値）からより外れにくくなり、符号化効率が向上する効果を奏する。

【0287】

逆量子化・逆変換部311は、エントロピー復号部301から入力された量子化変換係数をスケーリング（逆量子化）して変換係数 $d[][]$ を求める。この量子化変換係数は、符号化処理において、予測誤差に対してDCT（Discrete Cosine Transform、離散コサイン変換）、DST（Discrete Sine Transform、離散サイン変換）等の変換を行い量子化して得られる係数である。逆量子化・逆変換部311はtransform_skip_flagが0の場合、スケーリング済みの変換係数 $d[][]$ に対して、逆DCT、逆DST等の逆周波数変換を行い、予測誤差 $res[][]$ を算出する。逆量子化・逆変換部311はtransform_skip_flagが1の場合、 $res[x][y]=d[x][y]$ とする。逆量子化・逆変換部311は予測誤差を加算部312に出力する。

10

【0288】

なお、逆変換及び変換は、対になる処理であるため、変換と逆変換とを互いに置き換えて解釈してもよい。あるいは、逆変換を変換と呼ぶ場合には、変換を順変換と呼んでもよい。例えば、逆非分離変換を非分離変換と呼ぶ場合、非分離変換は順非分離変換と呼んでもよい。また、分離変換を単に変換と呼ぶ。

20

【0289】

加算部312は、予測画像生成部308から入力されたブロックの予測画像と逆量子化・逆変換部311から入力された予測誤差を画素毎に加算して、ブロックの復号画像を生成する。加算部312はブロックの復号画像を参照ピクチャメモリ306に記憶し、また、ループフィルタ305に出力する。

【0290】

（動画像符号化装置の構成）

次に、本実施形態に係る動画像符号化装置11の構成について説明する。図13は、本実施形態に係る動画像符号化装置11の構成を示すブロック図である。動画像符号化装置11は、予測画像生成部101、減算部102、変換・量子化部103、逆量子化・逆変換部105、加算部106、ループフィルタ107、予測パラメータメモリ（予測パラメータ記憶部、フレームメモリ）108、参照ピクチャメモリ（参照画像記憶部、フレームメモリ）109、符号化パラメータ決定部110、パラメータ符号化部111、エントロピー符号化部104を含んで構成される。

30

【0291】

予測画像生成部101は画像Tの各ピクチャを分割した領域であるCU毎に予測画像を生成する。予測画像生成部101は既に説明した予測画像生成部308と同じ動作であり、説明を省略する。

【0292】

減算部102は、予測画像生成部101から入力されたブロックの予測画像の画素値を、画像Tの画素値から減算して予測誤差を生成する。減算部102は予測誤差を変換・量子化部103に出力する。

40

【0293】

変換・量子化部103は、減算部102から入力された予測誤差に対し、周波数変換によって変換係数を算出し、量子化によって量子化変換係数を導出する。変換・量子化部103は、量子化変換係数をエントロピー符号化部104及び逆量子化・逆変換部105に出力する。

【0294】

逆量子化・逆変換部105は、動画像復号装置31における逆量子化・逆変換部311（図10

50

）と同じであり、説明を省略する。算出した予測誤差は加算部106に出力される。

【0295】

エントロピー符号化部104には、変換・量子化部103から量子化変換係数が入力され、パラメータ符号化部111から符号化パラメータが入力される。符号化パラメータは、例えば、予測モードを示すCuPredModeである。CuPredModeはイントラ予測を示すMODE_INTRA、インター予測を示すMODE_INTERのいずれでもよいし、MODE_INTRA、MODE_INTER、画面内のブロックをコピーして予測画像とするイントラブロックコピー予測を示すMODE_IBCであってもよい。

【0296】

エントロピー符号化部104は、分割情報、予測パラメータ、量子化変換係数等をエントロピー符号化して符号化ストリームTeを生成し、出力する。

【0297】

パラメータ符号化部111は、図示しないヘッダ符号化部1110、CT情報符号化部1111、CU符号化部1112（予測モード符号化部）、及びインター予測パラメータ符号化部112とイントラ予測パラメータ符号化部113を備えている。CU符号化部1112はさらにTU符号化部1114を備えている。

【0298】

以下、各モジュールの概略動作を説明する。パラメータ符号化部111はヘッダ情報、分割情報、予測情報、量子化変換係数等のパラメータの符号化処理を行う。

【0299】

CT情報符号化部1111は、符号化データからQT、MT（BT、TT）分割情報等を符号化する。

【0300】

CU符号化部1112はCU情報、予測情報、TU分割フラグ、CU残差フラグ等を符号化する。

【0301】

TU符号化部1114は、TUに予測誤差が含まれている場合に、QP更新情報と量子化変換係数を符号化する。

【0302】

CT情報符号化部1111、CU符号化部1112は、インター予測パラメータ、イントラ予測パラメータ、量子化変換係数等のシンタックス要素をエントロピー符号化部104に供給する。

【0303】

加算部106は、予測画像生成部101から入力されたブロックの予測画像の画素値と逆量子化・逆変換部105から入力された予測誤差を画素毎に加算して復号画像を生成する。加算部106は生成した復号画像を参照ピクチャメモリ109に記憶する。

【0304】

ループフィルタ107は加算部106が生成した復号画像に対し、デブロッキングフィルタ、SAO、ALFを施す。なお、ループフィルタ107は、必ずしも上記3種類のフィルタを含まなくてもよく、例えばデブロッキングフィルタのみの構成であってもよい。

【0305】

SAOは、サンプル単位で分類結果に応じたオフセットを加算するフィルタであり、ALFは、伝送したフィルタ係数と参照画像（もしくは参照画像と対象画素との差異）の積和を用いるフィルタである。

【0306】

予測パラメータメモリ108は、符号化パラメータ決定部110が生成した予測パラメータを、対象ピクチャ及びCU毎に予め定めた位置に記憶する。

【0307】

参照ピクチャメモリ109は、ループフィルタ107が生成した復号画像を対象ピクチャ及びCU毎に予め定めた位置に記憶する。

10

20

30

40

50

【0308】

符号化パラメータ決定部110は、符号化パラメータの複数のセットのうち、1つのセットを選択する。符号化パラメータとは、上述したQT、BTあるいはTT分割情報、予測パラメータ、あるいはこれらに関連して生成される符号化の対象となるパラメータである。予測画像生成部101は、これらの符号化パラメータを用いて予測画像を生成する。

【0309】

符号化パラメータ決定部110は、複数のセットの各々について情報量の大きさと符号化誤差を示すRDコスト値を算出する。符号化パラメータ決定部110は、算出したコスト値が最小となる符号化パラメータのセットを選択する。これにより、エントロピー符号化部104は、選択した符号化パラメータのセットを符号化ストリームTeとして出力する。符号化パラメータ決定部110は決定した符号化パラメータを予測パラメータメモリ108に記憶する。

【0310】

なお、上述した実施形態における動画像符号化装置11、動画像復号装置31の一部、例えば、エントロピー復号部301、パラメータ復号部302、ループフィルタ305、予測画像生成部308、逆量子化・逆変換部311、加算部312、予測画像生成部101、減算部102、変換・量子化部103、エントロピー符号化部104、逆量子化・逆変換部105、ループフィルタ107、符号化パラメータ決定部110、パラメータ符号化部111をコンピュータで実現するようにしてもよい。その場合、この制御機能を実現するためのプログラムをコンピュータ読み取り可能な記録媒体に記録して、この記録媒体に記録されたプログラムをコンピュータシステムに読み込ませ、実行することによって実現してもよい。なお、ここでいう「コンピュータシステム」とは、動画像符号化装置11、動画像復号装置31の何れかに内蔵されたコンピュータシステムであって、OSや周辺機器等のハードウェアを含むものとする。また、「コンピュータ読み取り可能な記録媒体」とは、フレキシブルディスク、光磁気ディスク、ROM、CD-ROM等の可搬媒体、コンピュータシステムに内蔵されるハードディスク等の記憶装置のことをいう。さらに「コンピュータ読み取り可能な記録媒体」とは、インターネット等のネットワークや電話回線等の通信回線を介してプログラムを送信する場合の通信線のように、短時間、動的にプログラムを保持するもの、その場合のサーバやクライアントとなるコンピュータシステム内部の揮発性メモリのように、一定時間プログラムを保持しているものも含んでもよい。また上記プログラムは、前述した機能の一部を実現するためのものであってもよく、さらに前述した機能をコンピュータシステムに既に記録されているプログラムとの組み合わせで実現できるものであってもよい。

【0311】

また、上述した実施形態における動画像符号化装置11、動画像復号装置31の一部、又は全部を、LSI (Large Scale Integration) 等の集積回路として実現してもよい。動画像符号化装置11、動画像復号装置31の各機能ブロックは個別にプロセッサ化してもよいし、一部、又は全部を集積してプロセッサ化してもよい。また、集積回路化の手法はLSIに限らず専用回路、又は汎用プロセッサで実現してもよい。また、半導体技術の進歩によりLSIに代替する集積回路化の技術が出現した場合、当該技術による集積回路を用いてもよい。

【0312】

以上、図面を参照してこの発明の一実施形態について詳しく説明してきたが、具体的な構成は上述のものに限られることはなく、この発明の要旨を逸脱しない範囲内において様々な設計変更等を行うことが可能である。

【0313】

〔応用例〕

上述した動画像符号化装置11及び動画像復号装置31は、動画像の送信、受信、記録、再生を行う各種装置に搭載して利用することができる。なお、動画像は、カメラ等により撮像された自然動画像であってもよいし、コンピュータ等により生成された人工動画像（C

10

20

30

40

50

G及びGUIを含む)であってもよい。

【0314】

まず、上述した動画像符号化装置11及び動画像復号装置31を、動画像の送信及び受信に利用できることを、図2を参照して説明する。

【0315】

図2には、動画像符号化装置11を搭載した送信装置PROD_Aの構成を示したブロック図が示されている。図に示すように、送信装置PROD_Aは、動画像を符号化することによって符号化データを得る符号化部PROD_A1と、符号化部PROD_A1が得た符号化データで搬送波を変調することによって変調信号を得る変調部PROD_A2と、変調部PROD_A2が得た変調信号を送信する送信部PROD_A3と、を備えている。上述した動画像符号化装置11は、この符号化部PROD_A1として利用される。

10

【0316】

送信装置PROD_Aは、符号化部PROD_A1に入力する動画像の供給源として、動画像を撮像するカメラPROD_A4、動画像を記録した記録媒体PROD_A5、動画像を外部から入力するための入力端子PROD_A6、及び、画像を生成又は加工する画像処理部A7を更に備えていてもよい。図においては、これら全てを送信装置PROD_Aが備えた構成を例示しているが、一部を省略しても構わない。

【0317】

なお、記録媒体PROD_A5は、符号化されていない動画像を記録したものであってもよいし、伝送用の符号化方式とは異なる記録用の符号化方式で符号化された動画像を記録したものであってもよい。後者の場合、記録媒体PROD_A5と符号化部PROD_A1との間に、記録媒体PROD_A5から読み出した符号化データを記録用の符号化方式に従って復号する復号部(不図示)を介在させるとよい。

20

【0318】

また、図2には、動画像復号装置31を搭載した受信装置PROD_Bの構成を示したブロック図が示されている。図に示すように、受信装置PROD_Bは、変調信号を受信する受信部PROD_B1と、受信部PROD_B1が受信した変調信号を復調することによって符号化データを得る復調部PROD_B2と、復調部PROD_B2が得た符号化データを復号することによって動画像を得る復号部PROD_B3と、を備えている。上述した動画像復号装置31は、この復号部PROD_B3として利用される。

30

【0319】

受信装置PROD_Bは、復号部PROD_B3が出力する動画像の供給先として、動画像を表示するディスプレイPROD_B4、動画像を記録するための記録媒体PROD_B5、及び、動画像を外部に出力するための出力端子PROD_B6を更に備えていてもよい。図においては、これら全てを受信装置PROD_Bが備えた構成を例示しているが、一部を省略しても構わない。

【0320】

なお、記録媒体PROD_B5は、符号化されていない動画像を記録するためのものであってもよいし、伝送用の符号化方式とは異なる記録用の符号化方式で符号化されたものであってもよい。後者の場合、復号部PROD_B3と記録媒体PROD_B5との間に、復号部PROD_B3から取得した動画像を記録用の符号化方式に従って符号化する符号化部(不図示)を介在させるとよい。

40

【0321】

なお、変調信号を伝送する伝送媒体は、無線であってもよいし、有線であってもよい。また、変調信号を伝送する伝送態様は、放送(ここでは、送信先が予め特定されていない送信態様を指す)であってもよいし、通信(ここでは、送信先が予め特定されている送信態様を指す)であってもよい。すなわち、変調信号の伝送は、無線放送、有線放送、無線通信、及び有線通信の何れによって実現してもよい。

【0322】

例えば、地上デジタル放送の放送局(放送設備等)／受信局(テレビジョン受像機等)は

50

、変調信号を無線放送で送受信する送信装置PROD_A／受信装置PROD_Bの一例である。また、ケーブルテレビ放送の放送局（放送設備等）／受信局（テレビジョン受像機等）は、変調信号を有線放送で送受信する送信装置PROD_A／受信装置PROD_Bの一例である。

【0323】

また、インターネットを用いたVOD（Video On Demand）サービスや動画共有サービス等のサーバ（ワークステーション等）／クライアント（テレビジョン受像機、パーソナルコンピュータ、スマートフォン等）は、変調信号を通信で送受信する送信装置PROD_A／受信装置PROD_Bの一例である（通常、LANにおいては伝送媒体として無線又は有線の何れかが用いられ、WANにおいては伝送媒体として有線が用いられる）。ここで、パーソナルコンピュータには、デスクトップ型PC、ラップトップ型PC、及びタブレット型PCが含まれる。また、スマートフォンには、多機能携帯電話端末も含まれる。

10

【0324】

なお、動画共有サービスのクライアントは、サーバからダウンロードした符号化データを復号してディスプレイに表示する機能に加え、カメラで撮像した動画像を符号化してサーバにアップロードする機能を有している。すなわち、動画共有サービスのクライアントは、送信装置PROD_A及び受信装置PROD_Bの双方として機能する。

【0325】

次に、上述した動画像符号化装置11及び動画像復号装置31を、動画像の記録及び再生に利用できることを、図3を参照して説明する。

20

【0326】

図3には、上述した動画像符号化装置11を搭載した記録装置PROD_Cの構成を示したブロック図が示されている。図に示すように、記録装置PROD_Cは、動画像を符号化することによって符号化データを得る符号化部PROD_C1と、符号化部PROD_C1が得た符号化データを記録媒体PROD_Mに書き込む書込部PROD_C2と、を備えている。上述した動画像符号化装置11は、この符号化部PROD_C1として利用される。

【0327】

なお、記録媒体PROD_Mは、（1）HDD（Hard Disk Drive）やSSD（Solid State Drive）等のように、記録装置PROD_Cに内蔵されるタイプのものであってもよいし、（2）SDメモリカードやUSB（Universal Serial Bus）フラッシュメモリ等のように、記録装置PROD_Cに接続されるタイプのものであってもよいし、（3）DVD（Digital Versatile Disc:登録商標）やBD（Blu-ray Disc:登録商標）等のように、記録装置PROD_Cに内蔵されたドライブ装置（不図示）に装填されるものであってもよい。

30

【0328】

また、記録装置PROD_Cは、符号化部PROD_C1に入力する動画像の供給源として、動画像を撮像するカメラPROD_C3、動画像を外部から入力するための入力端子PROD_C4、動画像を受信するための受信部PROD_C5、及び、画像を生成又は加工する画像処理部PROD_C6を更に備えていてもよい。図においては、これら全てを記録装置PROD_Cが備えた構成を例示しているが、一部を省略しても構わない。

【0329】

なお、受信部PROD_C5は、符号化されていない動画像を受信するものであってもよいし、記録用の符号化方式とは異なる伝送用の符号化方式で符号化された符号化データを受信するものであってもよい。後者の場合、受信部PROD_C5と符号化部PROD_C1との間に、伝送用の符号化方式で符号化された符号化データを復号する伝送用復号部（不図示）を介在させるとよい。

40

【0330】

このような記録装置PROD_Cとしては、例えば、DVDレコーダ、BDレコーダ、HDD（Hard Disk Drive）レコーダ等が挙げられる（この場合、入力端子PROD_C4又は受信部PROD_C5が動画像の主な供給源となる）。また、カムコーダ（この場合、カメラPROD_C3が動画像の主な供給源となる）、パーソナルコンピュータ（この場合、受信部PROD

50

_C5又は画像処理部C6が動画像の主な供給源となる）、スマートフォン（この場合、カメラPROD_C3又は受信部PROD_C5が動画像の主な供給源となる）等も、このような記録装置PROD_Cの一例である。

【0331】

また、図3には、上述した動画像復号装置31を搭載した再生装置PROD_Dの構成を示したブロック図が示されている。図に示すように、再生装置PROD_Dは、記録媒体PROD_Mに書き込まれた符号化データを読み出す読出部PROD_D1と、読出部PROD_D1が読み出した符号化データを復号することによって動画像を得る復号部PROD_D2と、を備えている。上述した動画像復号装置31は、この復号部PROD_D2として利用される。

【0332】

なお、記録媒体PROD_Mは、（1）HDDやSSD等のように、再生装置PROD_Dに内蔵されるタイプのものであってもよいし、（2）SDメモ리카ードやUSBフラッシュメモリ等のように、再生装置PROD_Dに接続されるタイプのものであってもよいし、（3）DVDやBD等のように、再生装置PROD_Dに内蔵されたドライブ装置（不図示）に装填されるものであってもよい。

【0333】

また、再生装置PROD_Dは、復号部PROD_D2が出力する動画像の供給先として、動画像を表示するディスプレイPROD_D3、動画像を外部に出力するための出力端子PROD_D4、及び、動画像を送信する送信部PROD_D5を更に備えていてもよい。図においては、これら全てを再生装置PROD_Dが備えた構成を例示しているが、一部を省略しても構わない。

【0334】

なお、送信部PROD_D5は、符号化されていない動画像を送信するものであってもよいし、記録用の符号化方式とは異なる伝送用の符号化方式で符号化された符号化データを送信するものであってもよい。後者の場合、復号部PROD_D2と送信部PROD_D5との間に、動画像を伝送用の符号化方式で符号化する符号化部（不図示）を介在させるとよい。

【0335】

このような再生装置PROD_Dとしては、例えば、DVDプレイヤー、BDプレイヤー、HDDプレイヤー等が挙げられる（この場合、テレビジョン受像機等が接続される出力端子PROD_D4が動画像の主な供給先となる）。また、テレビジョン受像機（この場合、ディスプレイPROD_D3が動画像の主な供給先となる）、デジタルサイネージ（電子看板や電子掲示板等とも称され、ディスプレイPROD_D3又は送信部PROD_D5が動画像の主な供給先となる）、デスクトップ型PC（この場合、出力端子PROD_D4又は送信部PROD_D5が動画像の主な供給先となる）、ラップトップ型又はタブレット型PC（この場合、ディスプレイPROD_D3又は送信部PROD_D5が動画像の主な供給先となる）、スマートフォン（この場合、ディスプレイPROD_D3又は送信部PROD_D5が動画像の主な供給先となる）等も、このような再生装置PROD_Dの一例である。

【0336】

（ハードウェアの実現及びソフトウェアの実現）

また、上述した動画像復号装置31及び動画像符号化装置11の各ブロックは、集積回路（ICチップ）上に形成された論理回路によってハードウェア的に実現してもよいし、CPU（Central Processing Unit）を用いてソフトウェア的に実現してもよい。

【0337】

後者の場合、上記各装置は、各機能を実現するプログラムの命令を実行するCPU、上記プログラムを格納したROM（Read Only Memory）、上記プログラムを展開するRAM（Random Access Memory）、上記プログラム及び各種データを格納するメモリ等の記憶装置（記録媒体）等を備えている。そして、本発明の実施形態の目的は、上述した機能を実現するソフトウェアである上記各装置の制御プログラムのプログラムコード（実行形式プログラム、中間コードプログラム、ソースプログラム）をコンピュータで読み取り可能に記録した記録媒体を、上記各装置に供給し、そのコンピュータ（又はCPUやMPU

10

20

30

40

50

）が記録媒体に記録されているプログラムコードを読み出し実行することによっても、達成可能である。

【0338】

上記記録媒体としては、例えば、磁気テープやカセットテープ等のテープ類、フロッピー（登録商標）ディスク／ハードディスク等の磁気ディスクやCD-ROM（Compact Disc Read-Only Memory）／MOディスク（Magneto-Optical disc）／MD（Mini Disc）／DVD（Digital Versatile Disc:登録商標）／CD-R（CD Recordable）／ブルーレイディスク（Blu-ray Disc:登録商標）等の光ディスクを含むディスク類、ICカード（メモリカードを含む）／光カード等のカード類、マスクROM／EPROM（Erasable Programmable Read-Only Memory）／EEPROM（Electrically Erasable and Programmable Read-Only Memory:登録商標）／フラッシュROM等の半導体メモリ類、あるいはPLD（Programmable logic device）やFPGA（Field Programmable Gate Array）等の論理回路類等を用いることができる。

10

【0339】

また、上記各装置を通信ネットワークと接続可能に構成し、上記プログラムコードを通信ネットワークを介して供給してもよい。この通信ネットワークは、プログラムコードを伝送可能であればよく、特に限定されない。例えば、インターネット、イントラネット、エキストラネット、LAN（Local Area Network）、ISDN（Integrated Services Digital Network）、VAN（Value-Added Network）、CATV（Community Antenna television/Cable Television）通信網、仮想専用網（Virtual Private Network）、電話回線網、移動体通信網、衛星通信網等が利用可能である。また、この通信ネットワークを構成する伝送媒体も、プログラムコードを伝送可能な媒体であればよく、特定の構成又は種類のものに限定されない。例えば、IEEE（Institute of Electrical and Electronic Engineers）1394、USB、電力線搬送、ケーブルTV回線、電話線、ADSL（Asymmetric Digital Subscriber Line）回線等の有線でも、IrDA（Infrared Data Association）やリモコンのような赤外線、Bluetooth（登録商標）、IEEE 802.11無線、HDR（High Data Rate）、NFC（Near Field Communication）、DLNA（Digital Living Network Alliance:登録商標）、携帯電話網、衛星回線、地上デジタル放送網等の無線でも利用可能である。なお、本発明の実施形態は、上記プログラムコードが電子的な伝送で具現化された、搬送波に埋め込まれたコンピュータデータ信号の形態でも実現され得る。

20

30

【0340】

本発明の実施形態は上述した実施形態に限定されるものではなく、請求項に示した範囲で種々の変更が可能である。すなわち、請求項に示した範囲で適宜変更した技術的手段を組み合わせて得られる実施形態についても本発明の技術的範囲に含まれる。

【産業上の利用可能性】

【0341】

本発明の実施形態は、画像データが符号化された符号化データを復号する動画像復号装置、及び、画像データが符号化された符号化データを生成する動画像符号化装置に好適に適用することができる。また、動画像符号化装置によって生成され、動画像復号装置によって参照される符号化データのデータ構造に好適に適用することができる。

40

【符号の説明】

【0342】

31 動画像復号装置

301 エントロピー復号部

302 パラメータ復号部

3020 ヘッダ復号部

308 予測画像生成部

311 逆量子化・逆変換部

312 加算部

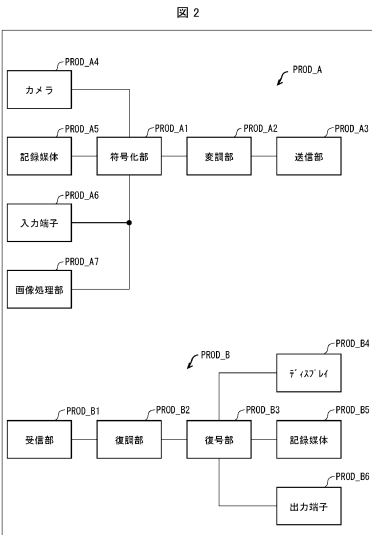
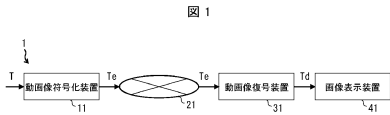
50

- 11 動画像符号化装置
 - 101 予測画像生成部
 - 102 減算部
 - 103 変換・量子化部
 - 104 エントロピー符号化部
 - 105 逆量子化・逆変換部
 - 107 ループフィルタ
 - 110 符号化パラメータ決定部
 - 111 パラメータ符号化部
 - 1110 ヘッダ符号化部
 - 1111 CT情報符号化部
 - 1112 CU符号化部（予測モード符号化部）
 - 1114 TU符号化部
 - 311 逆量子化・逆変換部
 - 3111 逆量子化部
 - 3112 逆変換部

【図面】

【図 1】

【図 2】



10

20

30

40

50

【圖 3】

【図 4】

图 3

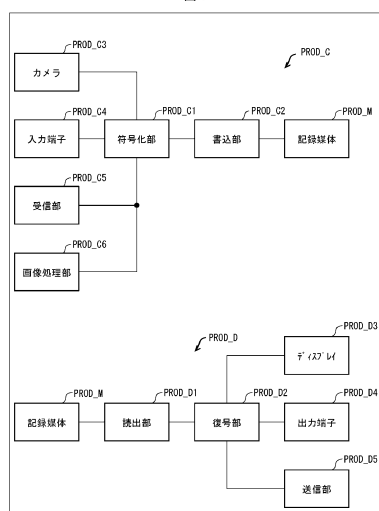
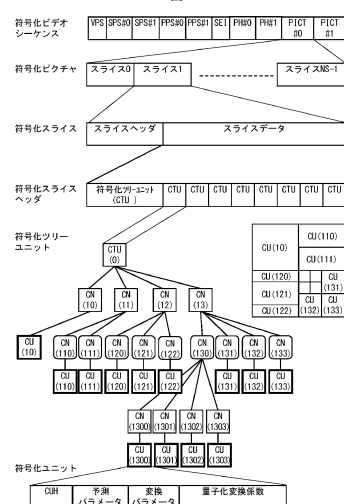


图 4



10

20

【図 5】

【図 6】

图 5

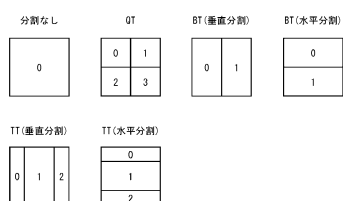
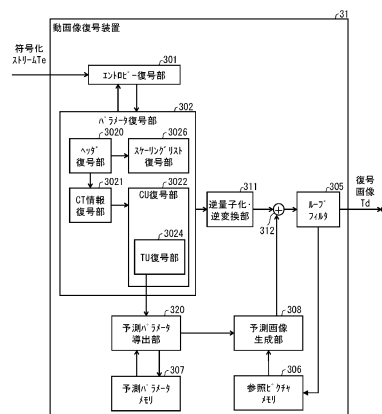


图 6

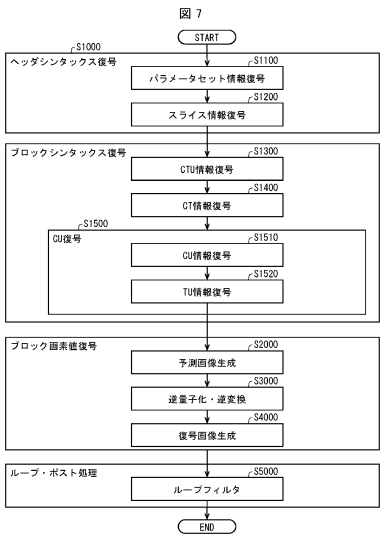


30

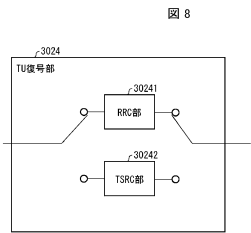
40

50

【図 7】



【図 8】



10

20

【図 9】

図 9

(a)	seq_parameter_set_rbsp() {	Descriptor
	sps_seq_parameter_set_id	u(4)
	sps_video_parameter_set_id	u(4)
	...	...
	sps_transform_skip_enabled_flag	u(1)
	if(sps_transform_skip_enabled_flag) {	
	sps_log2_transform_skip_max_size_minus2	ue(v)
	sps_bdpcm_enabled_flag	u(1)
	...	...
	if(sps_transform_skip_enabled_flag sps_palette_enabled_flag)	ue(v)
	sps_min_qp_prime_ts	...
	...	...
	sps_dep_quant_enabled_flag	u(1)
	sps_sign_data_hiding_enabled_flag	u(1)
	...	...
	}	

(b)	slice_header() {	Descriptor
	...	...
	if(sps_dep_quant_enabled_flag)	
	sh_dep_quant_used_flag	u(1)
	if(sps_sign_data_hiding_enabled_flag && !sh_dep_quant_used_flag)	u(1)
	sh_sign_data_hiding_used_flag	u(1)
	if(sps_transform_skip_enabled_flag && !sh_dep_quant_used_flag && !sh_sign_data_hiding_used_flag)	u(1)
	sh_ts_residual_coding_disabled_flag	u(1)
	...	...
	}	

【図 10】

図 10

(a)	coding_unit(x0, y0, cbWidth, cbHeight, cotDepth, treeType, modeType) {	Descriptor
	...	...
	if(sps_bdpcm_enabled_flag && cbWidth <= MaxTsSize && cbHeight <= MaxTsSize)	
	intra_bdpcm_luma_flag	ae(v)
	if(intra_bdpcm_luma_flag)	
	intra_bdpcm_luma_dir_flag	ae(v)
	else	
	...	ae(v)
	if(cbWidth/SubWidth <= MaxTsSize && cbHeight/SubHeight <= MaxTsSize	
	&& sps_bdpcm_enabled_flag)	
	intra_bdpcm_chroma_flag	ae(v)
	if(intra_bdpcm_chroma_flag)	
	intra_bdpcm_chroma_dir_flag	ae(v)
	else	
	...	...

(b)	transform_unit(x0, y0, tbWidth, tbHeight, treeType, subTindex, chType) {	Descriptor
	...	...
	if(!u.v.coded_flag[x0][y0] && treeType != DUAL_TREE_CHROMA) {	
	if(sps_transform_skip_enabled_flag && !BdpcmFlag[x0][y0][0] &&	
	tbWidth <= MaxTsSize && tbHeight <= MaxTsSize && (	
	IntraSubPartitionsSplitType == ISP_NO_SPLIT) && !u.sbt_flag)	
	transform_skip_flag[x0][y0][0]	ae(v)
	if(!transform_skip_flag[x0][y0][0])	
	residual_coding_disabled_flag	
	residual_coding(x0, y0, Log2(tbWidth), Log2(tbHeight), 0)	
	else	
	residual_ts_coding(x0, y0, Log2(tbWidth), Log2(tbHeight), 0)	
	}	
	...	...

30

40

50

【図 1 1 a】

図 11a

(a)	residual_coding(x0, y0, log2TbWidth, log2TbHeight, cldx) {	Descriptor
...	for (i = lastSubBlock; i >= 0; i--) {	
...	startQStateSb = QState	
...	xs = DiagScanOrder(log2TbWidth - log2SBW][log2TbHeight - log2SBH][i][0]	
...	ys = DiagScanOrder(log2TbWidth - log2SBW][log2TbHeight - log2SBH][i][1]	
...	inferSubSigCoeffFlag = 0	
...	if (i < lastSubBlock && i > 0) {	
...	sb_coded_flag[xs][ys]	ae(v)
...	inferSubSigCoeffFlag = 1	
...	...	
...	firstSigScanPosSb = numSbCoeff	
...	lastSigScanPosSb = -1	
...	...	
...	for (n = firstPosModel; n >= 0 && remBpass >= 4; n--) {	
...	x0 = (x0 << log2SBW) + DiagScanOrder(log2SBW][log2SBH][n][0]	
...	y0 = (y0 << log2SBH) + DiagScanOrder(log2SBW][log2SBH][n][1]	
...	if (sb_coded_flag[xs][ys] && (n > 0 !inferSubSigCoeffFlag) &&	
...	(x0 != LastSignificantCoeffX y0 != LastSignificantCoeffY) {	
...	sig_coeff_flag[x0][y0]	ae(v)
...	...	
...	if (sig_coeff_flag[x0][y0]) {	
...	abs_level_gtx_flag[n][0]	ae(v)
...	...	
...	if (abs_level_gtx_flag[n][0]) {	
...	par_level_flag[n]	ae(v)
...	...	
...	abs_level_gtx_flag[n][1]	ae(v)
...	...	
...	if (lastSigScanPosSb == -1)	
...	lastSigScanPosSb = n	
...	firstSigScanPosSb = n	
...	}	

【図 1 1 b】

図 11b

(b)	AbsLevelPass1[x0][y0] = sig_coeff_flag[x0][y0] + par_level_flag[n] +	
...	abs_level_gtx_flag[n][0] + 2 * abs_level_gtx_flag[n][1]	
...	if (sh_dep_quant_used_flag)	
...	QState = QStateFromTable(QState][AbsLevelPass1[x0][y0] & 1]	
...	firstPosModel = n - 1	
...	for (n = firstPosModel; n >= 0; n--) {	
...	x0 = (x0 << log2SBW) + DiagScanOrder(log2SBW][log2SBH][n][0]	
...	y0 = (y0 << log2SBH) + DiagScanOrder(log2SBW][log2SBH][n][1]	
...	if (abs_level_gtx_flag[n][1])	
...	abs_remainder[n]	ae(v)
...	AbsLevel[x0][y0] = AbsLevelPass1[x0][y0] + 2 * abs_remainder[n]	
...	for (n = firstPosModel; n >= 0; n--) {	
...	x0 = (x0 << log2SBW) + DiagScanOrder(log2SBW][log2SBH][n][0]	
...	y0 = (y0 << log2SBH) + DiagScanOrder(log2SBW][log2SBH][n][1]	
...	if (sb_coded_flag[xs][ys])	
...	dec_abs_level[n]	ae(v)
...	if (AbsLevel[x0][y0] > 0) {	
...	if (!lastSigScanPosSb == -1)	
...	lastSigScanPosSb = n	
...	if (sh_dep_quant_used_flag)	
...	QState = QStateFromTable(QState][AbsLevel[x0][y0] & 1]	
...	signifierFlag = sh_sign_data_flag_used_flag &&	
...	(!lastSigScanPosSb - firstSigScanPosSb > 3 ? 1 : 0)	
...	for (n = numSbCoeff - 1; n >= 0; n--) {	
...	x0 = (x0 << log2SBW) + DiagScanOrder(log2SBW][log2SBH][n][0]	
...	y0 = (y0 << log2SBH) + DiagScanOrder(log2SBW][log2SBH][n][1]	
...	if (AbsLevel[x0][y0] > 0) && (!signifierFlag (n != firstSigScanPosSb))	
...	coeff_sign_flag[n]	ae(v)
...	}	

10

20

【図 1 1 c】

図 11c

(c)	if (sh_dep_quant_used_flag) {	
...	QState = startQStateSb	
...	for (n = numSbCoeff - 1; n >= 0; n--) {	
...	x0 = (x0 << log2SBW) + DiagScanOrder(log2SBW][log2SBH][n][0]	
...	y0 = (y0 << log2SBH) + DiagScanOrder(log2SBW][log2SBH][n][1]	
...	if (AbsLevel[x0][y0] > 0)	← SYN101
...	TransCoeffLevel[x0][y0][cldx][x0][y0] =	
...	(2 * AbsLevel[x0][y0] - (QState > 1 ? 1 : 0)) +	
...	(1 - 2 * coeff_sign_flag[n])	
...	QState = QStateFromTable(QState][AbsLevel[x0][y0] & 1]	
...	} else {	
...	sumAbsLevel = 0	
...	for (n = numSbCoeff - 1; n >= 0; n--) {	
...	x0 = (x0 << log2SBW) + DiagScanOrder(log2SBW][log2SBH][n][0]	
...	y0 = (y0 << log2SBH) + DiagScanOrder(log2SBW][log2SBH][n][1]	
...	if (AbsLevel[x0][y0] > 0)	← SYN102
...	TransCoeffLevel[x0][y0][cldx][x0][y0] =	
...	AbsLevel[x0][y0] * (1 - 2 * coeff_sign_flag[n])	
...	if (!signifierFlag) {	
...	sumAbsLevel += AbsLevel[x0][y0]	
...	if (n == firstSigScanPosSb && sumAbsLevel % 2 == 1)	
...	TransCoeffLevel[x0][y0][cldx][x0][y0] =	
...	-TransCoeffLevel[x0][y0][cldx][x0][y0]	
...	}	
...	}	
...	}	

【図 1 2 a】

図 12a

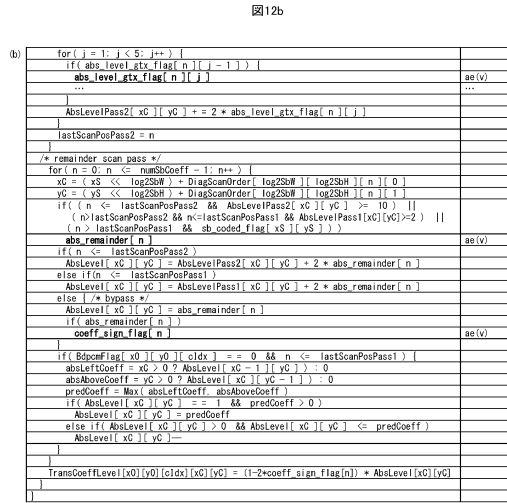
(a)	residual_ts_coding(x0, y0, log2TbWidth, log2TbHeight, cldx) {	Descriptor
...	for (i = 0; i <= lastSubBlock; i++) {	
...	xs = DiagScanOrder(log2TbWidth - log2SBW][log2TbHeight - log2SBH][i][0]	
...	ys = DiagScanOrder(log2TbWidth - log2SBW][log2TbHeight - log2SBH][i][1]	
...	if (i != lastSubBlock !inferSubCof)	
...	coded_sub_block_flag[xs][ys]	ae(v)
...	/* First scan pass */	
...	...	
...	lastScanPosPass1 = -1	
...	for (n = 0; n <= numSbCoeff - 1 && RemCobs >= 4; n++) {	
...	x0 = (x0 << log2SBW) + DiagScanOrder(log2SBW][log2SBH][n][0]	
...	y0 = (y0 << log2SBH) + DiagScanOrder(log2SBW][log2SBH][n][1]	
...	lastScanPosPass1 = n	
...	if (sb_coded_flag[xs][ys] && (n != numSbCoeff - 1 !inferSubSigCoeffFlag))	
...	sig_coeff_flag[x0][y0]	ae(v)
...	...	
...	if (sig_coeff_flag[x0][y0])	
...	coeff_sign_flag[n]	ae(v)
...	...	
...	abs_level_gtx_flag[n][0]	ae(v)
...	...	
...	if (abs_level_gtx_flag[n][0]) {	
...	par_level_flag[n]	ae(v)
...	...	
...	AbsLevelPass1[x0][y0] =	
...	sig_coeff_flag[x0][y0] + par_level_flag[n] + abs_level_gtx_flag[n][0]	
...	/* Greater than X scan pass (numUXFlags=5) */	
...	lastScanPosPass2 = -1	
...	for (n = 0; n <= numSbCoeff - 1 && RemCobs >= 4; n++) {	
...	x0 = (x0 << log2SBW) + DiagScanOrder(log2SBW][log2SBH][n][0]	
...	y0 = (y0 << log2SBH) + DiagScanOrder(log2SBW][log2SBH][n][1]	
...	AbsLevelPass2[x0][y0] = AbsLevelPass1[x0][y0]	

30

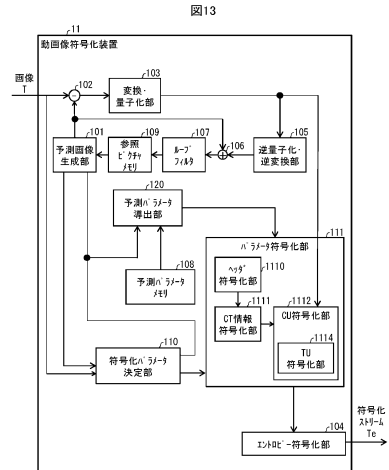
40

50

【図 1 2 b】



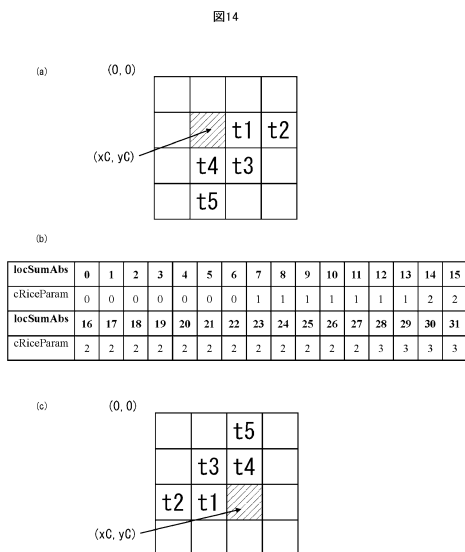
【図 1 3】



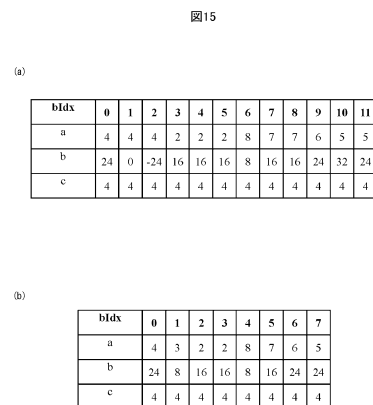
10

20

【図 1 4】



【図 1 5】



30

40

50

【図 16】

図16

(a)

cRiceParam = floor(log2((val * parA)>>parC)) - parB

(b)

cRiceParam = floor(log2((val * parA))) - parB

【図 17】

図17

(a)

cRiceParam = floor(log2((parA * locSumAbs + parB) >> parC)) + parE

(b)

cRiceParam = floor(log2((parA * locSumAbs + parB)) + parE

(c)

cRiceParam = floor(log2((locSumAbs<<parS) + parB)) + parE

(d)

cRiceParam = floor(log2((parA * locSumAbs + parB + parD * remstat) >> parC)) + parE

(e)

cRiceParam = floor(log2((parA * locSumAbs + parB + parD * remstat)) + parE

(f)

cRiceParam = floor(log2((locSumAbs<<parS) + parB + (remstat<<parT))) + parE

10

20

【図 18】

図18

(a)

slice_header_extension() {	Descriptor
if (rice_adaptation_enabled_flag)	
{	
slice_rice_paramA_minus1	us(4)
slice_rice_paramB	us(4)
slice_rice_paramC	us(4)
slice_rice_paramD	us(4)
}	
}	

(b)

slice_header_extension() {	Descriptor
if (rice_adaptation_enabled_flag)	
{	
slice_rice_rem_paramA_minus1	us(4)
slice_rice_rem_paramB	us(4)
slice_rice_rem_paramD	us(4)
slice_rice_abs_paramA_minus1	us(4)
slice_rice_abs_paramB	us(4)
slice_rice_abs_paramD	us(4)
}	
}	

(c)

slice_header_extension() {	Descriptor
if (rice_adaptation_enabled_flag)	
{	
slice_rice_rem_paramS_minus1	us(4)
slice_rice_rem_paramB	us(4)
slice_rice_rem_paramD	us(4)
slice_rice_abs_paramS_minus1	us(4)
slice_rice_abs_paramB	us(4)
slice_rice_abs_paramD	us(4)
}	
}	

30

40

50

フロントページの続き

(32)優先日 令和2年11月26日(2020.11.26)

(33)優先権主張国・地域又は機関
日本国(JP)

(31)優先権主張番号 特願2020-199372(P2020-199372)

(32)優先日 令和2年12月1日(2020.12.1)

(33)優先権主張国・地域又は機関
日本国(JP)

(31)優先権主張番号 特願2020-213074(P2020-213074)

(32)優先日 令和2年12月23日(2020.12.23)

(33)優先権主張国・地域又は機関
日本国(JP)

大阪府堺市堺区匠町1番地 シャープ株式会社内

Fターム(参考) 5C159 LC09 MA04 MA05 MA21 MC11 ME01 RC12 TA57 TB08 TC04
TC26 TC42 UA02 UA04 UA16 UA21