

(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7701981号
(P7701981)

(45)発行日 令和7年7月2日(2025.7.2)

(24)登録日 令和7年6月24日(2025.6.24)

(51)国際特許分類

F I

H O 4 N 19/70 (2014.01)

H O 4 N 19/70

H O 4 N 19/91 (2014.01)

H O 4 N 19/91

請求項の数 6 (全123頁)

(21)出願番号	特願2023-540955(P2023-540955)	(73)特許権者	521024075
(86)(22)出願日	令和4年1月4日(2022.1.4)		ベイジン・ダジャ・インターネット・イ
(65)公表番号	特表2024-502109(P2024-502109		ンフォメーション・テクノロジー・カン
	A)		パニー, リミテッド
(43)公表日	令和6年1月17日(2024.1.17)		中華人民共和国 1 0 0 0 8 5 ベイジン
(86)国際出願番号	PCT/US2022/011183		, ハイディアン・ディストリクト, シャ
(87)国際公開番号	WO2022/147570		ンディ・ウエスト・ロード, ナンバー
(87)国際公開日	令和4年7月7日(2022.7.7)		6, ビルディング 1, ファースト・フ
審査請求日	令和5年7月4日(2023.7.4)		ロア, ルーム 1 0 1 ディー1-7
(31)優先権主張番号	63/133,765	(74)代理人	100118902
(32)優先日	令和3年1月4日(2021.1.4)		弁理士 山本 修
(33)優先権主張国・地域又は機関	米国(US)	(74)代理人	100106208
			弁理士 宮前 徹
		(74)代理人	100196508
			弁理士 松尾 淳一

最終頁に続く

(54)【発明の名称】 ビデオコーディング用の残差および係数のコーディング

(57)【特許請求の範囲】

【請求項1】

ビデオコーディングのための方法であって、
デコーダによって、シーケンスパラメータセット（SPS）を参照するスライスヘッダ（SH）シンタックス構造内に変換スキップ残差コーディングのシンタックス構造に使用されるシンタックス要素が存在するかどうかを示すSPS残差符号化フラグを受信するステップと、
前記SPS残差符号化フラグの値が1に等しいとの決定に応答して、前記SPSを参照する前記SHシンタックス構造内に前記シンタックス要素が存在すると決定するステップと、
前記SPS残差符号化フラグの前記値が0に等しいとの決定に応答して、前記SPSを参照する前記SHシンタックス構造内に前記シンタックス要素が存在しないと決定するステップと、
前記シンタックス要素が存在しないときは、当該シンタックス要素の値が0に等しいと推定するステップと
を含む方法。

【請求項2】

前記デコーダによって、インデックスpps__ts__residual__coding__rice__indexが前記ビデオの現在のピクチャに存在するかどうかを示すピクチャパラメータセット（PPS）残差符号化フラグを受信するステップと、

前記 P P S 残差符号化ライスフラグの値が 1 に等しいとの決定に応答して、前記 p p s _ t s _ r e s i d u a l _ c o d i n g _ r i c e _ i n d e x が前記現在のピクチャに存在すると決定するステップと、

前記 P P S 残差符号化フラグの前記値が 0 に等しいとの決定に応答して、前記 p p s _ t s _ r e s i d u a l _ c o d i n g _ r i c e _ i n d e x が前記現在のピクチャに存在しないと決定するステップと

をさらに含む、請求項 1 に記載のビデオコーディングのための方法。

【請求項 3】

前記 S P S 残差符号化フラグの値が 0 に等しいときは、変換スキップ残差符号化ライス制約フラグの値が 1 に等しくなるように設定するステップをさらに含む、請求項 1 に記載のビデオコーディングのための方法。

【請求項 4】

ビデオコーディングのための装置であって、

1 つまたは複数のプロセッサと、

前記 1 つまたは複数のプロセッサによって実行可能な命令を記憶するように構成されたメモリとを含み、

前記 1 つまたは複数のプロセッサは、前記命令の実行時に、請求項 1 から 3 のいずれかに記載の方法を実行するように構成される、装置。

【請求項 5】

コンピュータ実行可能命令を記憶している、ビデオコーディングのための非一時的なコンピュータ可読記憶媒体であって、前記コンピュータ実行可能命令は、1 つまたは複数のコンピュータプロセッサによって実行されると、ビデオビットストリームを処理して当該処理されたビデオビットストリームを前記非一時的なコンピュータ可読記憶媒体に記憶させるために前記 1 つまたは複数のコンピュータプロセッサに請求項 1 から 3 のいずれかに記載の方法を実行させる、ビデオコーディングのための非一時的なコンピュータ可読記憶媒体。

【請求項 6】

1 つまたは複数のプロセッサを有するコンピューティングデバイスによる実行のための命令を含むコンピュータプログラムであって、前記命令は、前記 1 つまたは複数のプロセッサによって実行されると、前記コンピューティングデバイスに請求項 1 から 3 のいずれかに記載の方法を実行させる、コンピュータプログラム。

【発明の詳細な説明】

【技術分野】

【0001】

関連出願への相互参照

[0001]本出願は、2021年1月4日に提出された仮出願第63／133，765号に基づき優先権を主張し、その全内容はあらゆる目的で参照により本明細書に組み込まれる。

【0002】

[0002]本開示は、ビデオコーディングおよび圧縮に関する。より詳細には、本開示は、ビデオコーディングのための残差および係数のコーディングの改善および簡素化に関する。

【背景技術】

【0003】

[0003]ビデオデータを圧縮するために、さまざまなビデオコーディング技術が使用される。ビデオコーディングは、1 つまたは複数のビデオコーディング規格に従って実行される。例えば、ビデオコーディング規格は、多用途ビデオコーディング (VVC)、共同探索テストモデル (JEM)、高効率ビデオコーディング (H.265 / HEVC)、アドバンスドビデオコーディング (H.264 / AVC)、ムービングピクチャエキスパートグループ (MPEG) コーディングなどを含む。ビデオコーディングは一般に、ビデオ画像またはシーケンスに存在する冗長性を利用する予測方法 (例えば、インター予測、イントラ予測など) を利用する。ビデオコーディング技術の重要な目標は、ビデオ品質の低

10

20

30

40

50

下を回避または最小限に抑えながら、ビデオデータをより低いビットレートを使用する形式に圧縮することである。

【発明の概要】

【0004】

[0004]本開示の例は、ビデオコーディングのための方法および装置を提供する。

【0005】

[0005]本開示の第1の態様によれば、ビデオコーディングのための方法が提供される。この方法は、デコーダ（復号器）によって、シーケンスパラメータセット（SPS）を参照するSHシンタックス構造内にインデックス `sh__ts__residual__coding__rice__idx` が存在するかどうかを示すSPS残差符号化フラグを受信することと、SPS残差符号化フラグの値が1に等しいとの決定にตอบสนองして、SPSを参照するスライスヘッド（SH）シンタックス構造内に `sh__ts__residual__coding__rice__idx` が存在すると決定することと、残差符号化フラグの値が0に等しいとの決定にตอบสนองして、SPSを参照するSHシンタックス構造内に `sh__ts__residual__coding__rice__idx` が存在しないと決定することと、を含み得る。

10

【0006】

[0006]本開示の第2の態様によれば、ビデオコーディングのための方法が提供される。この方法は、デコーダによって、シンタックスである `abs__remaining` および `dec__abs__level` の2値化のための代替ライスパラメータ導出が使用されるかどうかを示すシーケンスパラメータセット（SPS）適応可能フラグを受信することと、SPS適応可能フラグの値が1に等しいとの決定にตอบสนองして、シンタックスの2値化のための代替ライスパラメータ導出が使用されると決定することと、SPS適応可能フラグの値が0に等しいとの決定にตอบสนองして、シンタックスの2値化のための代替ライスパラメータ導出が使用されないと決定することと、を含み得る。

20

【0007】

[0007]本開示の第3の態様によれば、ビデオコーディングのための方法が提供される。この方法は、デコーダによって、他のフラグに対する一般的な制約制御を提供するために残差符号化ライス制約フラグを受信することと、残差符号化ライス制約フラグの値が1に等しいとの決定にตอบสนองして、他のフラグの値が0に等しいと決定することと、を含み得る。

30

【0008】

[0008]上記の一般的な説明および以下の詳細な説明は、単なる例示および説明にすぎず、本開示を限定することを意図したものではないことを理解されたい。

【0009】

[0009]本明細書に組み込まれ、その一部を構成する添付図面は、本開示と一致する例を示し、説明とともに本開示の原理を説明するのに役立つ。

【図面の簡単な説明】

【0010】

【図1】[0010]本開示の一例によるエンコーダ（符号化器）のブロック図である。

【図2】[0011]本開示の一例によるデコーダ（復号器）のブロック図である。

40

【図3A】[0012]本開示の一例による、多重ツリー構造におけるブロック分割を示す図である。

【図3B】[0013]本開示の一例による、多重ツリー構造におけるブロック分割を示す図である。

【図3C】[0014]本開示の一例による、多重ツリー構造におけるブロック分割を示す図である。

【図3D】[0015]本開示の一例による、多重ツリー構造におけるブロック分割を示す図である。

【図3E】[0016]本開示の一例による、多重ツリー構造におけるブロック分割を示す図である。

50

【図 4】[0017]本開示の一例による、 18×12 輝度 C T U を有するピクチャを示す図である。

【図 5】[0018]本開示の一例による、 18×12 輝度 C T U を有するピクチャの図である。

【図 6 A】[0019]本開示の一例による、V T M における許可されない 3 値ツリー (T T) および 2 値ツリー (B T) 分割の一例を示す図である。

【図 6 B】[0020]本開示の一例による、V T M における許可されない T T および B T 分割の一例を示す図である。

【図 6 C】[0021]本開示の一例による、V T M における許可されない T T および B T 分割の一例を示す図である。

10

【図 6 D】[0022]本開示の一例による、V T M における許可されない T T および B T 分割の一例を示す図である。

【図 6 E】[0023]本開示の一例による、V T M における許可されない T T および B T 分割の一例を示す図である。

【図 6 F】[0024]本開示の一例による、V T M における許可されない T T および B T 分割の一例を示す図である。

【図 6 G】[0025]本開示の一例による、V T M における許可されない T T および B T 分割の一例を示す図である。

【図 6 H】[0026]本開示の一例による、V T M における許可されない T T および B T 分割の一例を示す図である。

20

【図 7】[0027]本開示の一例による、変換ブロックの残差符号化構造を示す図である。

【図 8】[0028]本開示の一例による、変換スキップブロックの残差符号化構造を示す図である。

【図 9】[0029]本開示の一例による、2 つのスカラー量子化器を示す図である。

【図 10 A】[0030]本開示の一例による状態遷移を示す図である。

【図 10 B】[0031]本開示の一例による量子化器の選択を示す図である。

【図 11】[0032]本開示による、確率モデルを選択するために使用されるテンプレートの図である。

【図 12】[0033]本開示による、パレットモードで符号化されたブロックの一例を示す図である。

30

【図 13】[0034]本開示による、パレットエントリを伝送するためのパレット予測子の使用を示す図である。

【図 14 A】[0035]本開示による水平トラバーススキャンを示す図である。

【図 14 B】[0036]本開示による垂直トラバーススキャンを示す図である。

【図 15 A】[0037]本開示による、パレットをスキャンするサブブロックベースのインデックスマップを示す図である。

【図 15 B】[0038]本開示による、パレットをスキャンするサブブロックベースのインデックスマップを示す図である。

【図 16】[0039]本開示の一例によるビデオ信号を符号化する方法の図である。

【図 17】[0040]本開示の一例によるビデオ信号を符号化する方法の図である。

40

【図 18】[0041]本開示の一例による、ユーザインターフェースと結合されたコンピューティング環境を示す図である。

【図 19】[0042]本開示の一例による、ビデオコーディングの方法を示す図である。

【図 20】[0043]本開示の一例による、ビデオコーディングの方法を示す図である。

【図 21】[0044]本開示の一例による、ビデオコーディングの方法を示す図である。

【発明を実施するための形態】

【0011】

[0045]次に、例示的な実施形態を詳細に参照し、その例を添付の図面に示す。以下の説明は添付の図面を参照しており、別段の記載がない限り、異なる図面における同じ番号は同じまたは類似の要素を表す。例示的な実施形態の以下の説明に記載される実装は、本開

50

示と一致するすべての実装を表すものではない。むしろ、それらは、添付の特許請求の範囲に記載される本開示に関連する態様と一致する装置および方法の単なる例にすぎない。

【0012】

【0046】本開示で使用される用語は、特定の実施形態を説明することのみを目的としており、本開示を限定することを意図したものではない。本開示および添付の特許請求の範囲で使用される場合、単数形「a」、「an」、および「the」は、文脈上明らかに別段の指示がない限り、複数形も含むものとする。本明細書で使用される用語「および／または」は、関連する列挙された項目の1つまたは複数の任意またはすべての可能な組み合わせを意味し、含むことを意図していることも理解されたい。

【0013】

【0047】「第1」、「第2」、「第3」などの用語は、本明細書ではさまざまな情報を説明するために使用され得るが、情報はこれらの用語によって限定されるべきではないことを理解されたい。これらの用語は、情報の1つのカテゴリを別のカテゴリから区別するためにのみ使用される。例えば、本開示の範囲から逸脱することなく、第1の情報を第2の情報と呼ぶこともできる。同様に、第2の情報を第1の情報と呼ぶこともできる。本明細書で使用される「場合」という用語は、文脈に応じて、「いつ」、「その際に」、または「判断に応じて」を意味すると理解され得る。

【0014】

【0048】HEVC規格の第1のバージョンは2013年10月に完成し、前世代のビデオコーディング規格H.264/MPEG AVCと比較して約50%のビットレート節約または同等の知覚品質を提供する。HEVC規格では、以前の規格に比べてコーディングが大幅に改善されているが、追加のコーディングツールを使用すると、HEVCよりも優れた符号化効率達成できるという証拠がある。それに基づいて、VCEGとMPEGの両社は、将来のビデオコーディング規格化に向けた新しいコーディング技術の探索作業を開始し、2015年10月にITU-T VCEGとISO/IEC MPEGによって共同ビデオ探索チーム(JVET)が結成され、符号化効率の大幅な向上を可能にし得る先進技術の重要な研究を開始した。共同探索モデル(JEM)と呼ばれる1つの参照ソフトウェアは、HEVCテストモデル(HM)の上いくつかの追加のコーディングツールを統合することによってJVETによって維持された。

【0015】

【0049】2017年10月に、HEVCを超える機能を備えたビデオ圧縮に関する共同提案募集(CFP)がITU-TおよびISO/IECによって発行された。2018年4月、第10回JVET会議で23件のCFP回答が受領され、評価され、HEVCよりも約40%圧縮効率が向上することが実証された。このような評価結果に基づいて、JVETはVersatile Video Coding(VVC)と名付けられた新世代のビデオコーディング規格を開発する新しいプロジェクトを立ち上げた。同月、VVCテストモデル(VTM)と呼ばれる1つの参照ソフトウェアコードベースが、VVC規格の参照実装を実証するために確立された。

【0016】

【0050】HEVCと同様に、VVCは、ブロックベースのハイブリッド型ビデオコーディングのフレームワークに基づいて構築される。

【0017】

【0051】図1は、VVC用のブロックベースのビデオエンコーダ(ビデオ符号化器)の一般的な図を示す。具体的には、図1は、典型的なエンコーダ100を示す。エンコーダ100は、ビデオ入力110、動き補償112、動き推定114、イントラ/インターモード決定116、ブロック予測子140、加算器128、変換130、量子化132、予測関連情報142、イントラ予測118、ピクチャバッファ120、逆量子化134、逆変換136、加算器126、メモリ124、ループ内フィルタ122、エントロピー符号化138、およびビットストリーム144を有する。

【0018】

[0052]エンコーダ100では、ビデオフレームは処理のために複数のビデオブロックに分割される。所定のビデオブロックごとに、インター予測アプローチまたはイントラ予測アプローチのいずれかに基づいて予測が形成される。

【0019】

[0053]ビデオ入力110の一部である現在のビデオブロックと、ブロック予測子140の一部であるその予測子との間の差を表す予測残差は、加算器128から変換130に送信される。次に、変換係数は、エントロピー削減のために変換130から量子化132に送られる。次いで、量子化された係数がエントロピー符号化138に供給されて、圧縮ビデオビットストリームが生成される。図1に示すように、ビデオブロック分割情報、動きベクトル(MV)、参照ピクチャインデックス、およびイントラ予測モードなどの、イントラ／インターモード決定116からの予測関連情報142はまた、エントロピー符号化138を通じて供給され、圧縮ビットストリーム144に保存される。圧縮ビットストリーム144はビデオビットストリームを含む。

【0020】

[0054]エンコーダ100では、予測の目的でピクセルを再構成するためにデコーダ(復号器)関連回路も必要である。逆量子化134および逆変換136を通じて予測残差が再構成される。この再構成された予測残差は、ブロック予測器140と結合されて、現在のビデオブロックのフィルタリングされていない再構成ピクセルを生成する。

【0021】

[0055]空間予測(または「イントラ予測」)は、現在のビデオブロックと同じビデオフレーム内の既に符号化された隣接ブロックのサンプル(参照サンプルと呼ばれる)からのピクセルを使用して、現在のビデオブロックを予測する。

【0022】

[0056]時間予測(「インター予測」とも呼ばれる)は、既に符号化されたビデオピクチャから再構成されたピクセルを使用して、現在のビデオブロックを予測する。時間予測は、ビデオ信号に固有の時間的冗長性を軽減する。特定の符号化ユニット(CU)または符号化ブロックの時間予測信号は、通常、現在のCUとその時間参照の間の動きの量と方向を示す1つまたは複数のMVによって伝送される。さらに、複数の参照ピクチャがサポートされている場合、時間予測信号が参照ピクチャストレージ内のどの参照ピクチャからのものであるかを識別するために使用される、1つの参照ピクチャインデックスが追加で送信される。

【0023】

[0057]動き推定114は、ビデオ入力110およびピクチャバッファ120からの信号を取り込み、動き補償112に動き推定信号を出力する。動き補償112は、ビデオ入力110、画像バッファ120からの信号、および動き推定114からの動き推定信号を取り込み、動き補償信号をイントラ／インターモード決定116に出力する。

【0024】

[0058]空間的および／または時間的予測が実行された後、エンコーダ100内のイントラ／インターモード決定116は、例えば定格歪み最適化方法に基づいて最良の予測モードを選択する。次いで、ブロック予測子140が現在のビデオブロックから減算され、その結果得られる予測残差が、変換130および量子化132を使用して相関解除される。結果として得られる量子化残差係数は、逆量子化134によって逆量子化され、逆変換136によって逆変換されて再構成残差が形成され、その後、予測ブロックに戻って加算されてCUの再構成信号が形成される。デブロッキングフィルタ、サンプル適応オフセット(SAO)、および／または適応ループ内フィルタ(ALF)などのさらなるループ内フィルタリング122は、再構成されたCUがピクチャバッファ120の参照ピクチャストレージに入れられ、将来のビデオブロックを符号化するために使用される前に、再構成されたCUに適用することができる。出力ビデオビットストリーム144を形成するために、符号化モード(インターまたはイントラ)、予測モード情報、動き情報、および量子化された残差係数はすべてエントロピー符号化ユニット138に送られ、さらに圧縮および

10

20

30

40

50

パックされてビットストリームが形成される。

【0025】

【0059】図1は、一般的なブロックベースのハイブリッド型ビデオコーディングシステムのブロック図を示す。入力ビデオ信号はブロックごとに処理される（符号化ユニット（CU）と呼ばれる）。VTM-1.0では、CUは最大 128×128 ピクセルになり得る。しかし、4値ツリーのみに基づいてブロックを分割するHEVCとは異なり、VVCでは1つの符号化ツリーユニット（CTU）がCUに分割され、4値／2値／3値ツリーに基づいてさまざまな局所特性に適應する。定義により、符号化ツリーブロック（CTB）は、成分をCTBに分割することが分割となる、Nのある値に対するサンプルの $N \times N$ ブロックである。CTUには、輝度（luma）サンプルのCTB、3つのサンプル配列を持つピクチャの色差（chroma）サンプルの2つの対応するCTB、またはモノクロピクチャもしくはサンプルの符号化に使用される3つの別個のカラープレーンとシンタックス構造を使用して符号化されるピクチャのサンプルのCTBが含まれる。さらに、HEVCにおける複数の分割ユニットタイプの概念が削除される。つまり、CU、予測ユニット（PU）、および変換ユニット（TU）の分離はVVCにはもう存在しない。代わりに、各CUは、さらに分割することなく、予測と変換の両方の基本単位として常に使用される。多重ツリー構造では、まず、1つのCTUが4値ツリー構造で分割される。次に、各4値ツリーリーフノードは、2値ツリーおよび3値ツリー構造によってさらに分割できる。図3A、3B、3C、3D、および3Eに示すように、4分割、水平2分割、垂直2分割、水平3分割、および垂直3分割の5つの分割タイプがある。

10

20

【0026】

【0060】図3Aは、本開示による、多重ツリー構造におけるブロックの4分割を表す図を示している。

【0027】

【0061】図3Bは、本開示による、多重ツリー構造におけるブロック垂直2分割を表す図を示している。

【0028】

【0062】図3Cは、本開示による、多重ツリー構造におけるブロック水平2分割を表す図を示している。

【0029】

【0063】図3Dは、本開示による、多重ツリー構造におけるブロック垂直3分割を表す図を示している。

【0030】

【0064】図3Eは、本開示による、多重ツリー構造におけるブロック水平3分割を表す図を示している。

【0031】

【0065】図1において、空間予測および／または時間予測を実行することができる。空間予測（または「イントラ予測」）は、同じビデオピクチャ／スライス内の既に符号化された隣接ブロックのサンプル（参照サンプルと呼ばれる）のピクセルを使用して、現在のビデオブロックを予測する。空間予測は、ビデオ信号に固有の空間的冗長性を軽減する。時間予測（「インター予測」または「動き補償予測」とも呼ばれる）は、既に符号化されたビデオピクチャから再構成されたピクセルを使用して、現在のビデオブロックを予測する。時間予測は、ビデオ信号に固有の時間的冗長性を軽減する。特定のCUの時間予測信号は、通常、現在のCUとその時間参照の間の動きの量と方向を示す1つまたは複数の動きベクトル（MV）によって伝送される。また複数の参照ピクチャがサポートされている場合、時間予測信号が参照ピクチャストア内のどの参照ピクチャから来たのかを識別するために使用される、1つの参照ピクチャインデックスが追加で送信される。空間的および／または時間的予測の後、エンコーダ内のモード決定ブロックは、例えば定格歪み最適化方法に基づいて、最適な予測モードを選択する。次に、予測ブロックが現在のビデオブロックから減算される。予測残差は、変換を使用して相関が解除され、量子化される。量子化

30

40

50

された残差係数は逆量子化され、逆変換されて再構成された残差が形成され、その後、予測ブロックに戻って加算されてC Uの再構成された信号が形成される。デブロッキングフィルタ、サンプル適応オフセット（S A O）、および適応ループ内フィルタ（A L F）などのさらなるループ内フィルタ処理を、再構成されたC Uが参照ピクチャストアに置かれ、将来のビデオブロックの符号化に使用される前に、再構成されたC Uに適用することができる。出力ビデオビットストリームを形成するには、符号化モード（インタ—またはイントラ）、予測モード情報、動き情報、および量子化された残差係数がすべてエントロピー符号化ユニットに送信され、さらに圧縮およびパックされてビットストリームが形成される。

【0032】

【0066】図2は、V V C用のビデオデコーダ（ビデオ復号器）の一般的なブロック図を示す。具体的には、図2は、典型的なデコーダ200のブロック図を示す。デコーダ200は、ビットストリーム210、エントロピー復号212、逆量子化214、逆変換216、加算器218、イントラ／インターモード選択220、イントラ予測222、メモリ230、ループ内フィルタ228、動き補償224、ピクチャバッファ226、予測関連情報234、およびビデオ出力232を有する。

【0033】

【0067】デコーダ200は、図1のエンコーダ100内に存在する再構成関連セクションと同様である。デコーダ200では、入力ビデオビットストリーム210がまずエントロピー復号212を通じて復号され、量子化された係数レベルおよび予測関連情報が導出される。次に、量子化された係数レベルは、逆量子化214および逆変換216を通じて処理されて、再構成された予測残差が得られる。イントラ／インターモードセクタ220に実装されるブロック予測機構は、復号された予測情報に基づいて、イントラ予測222または動き補償224のいずれかを実行するように構成される。フィルタリングされていない再構成ピクセルのセットは、加算器218を使用して、逆変換216からの再構成予測残差と、ブロック予測機構によって生成された予測出力とを合計することによって得られる。

【0034】

【0068】再構成されたブロックは、参照ピクチャストアとして機能するピクチャバッファ226に記憶される前に、ループ内フィルタ228をさらに通過することができる。ピクチャバッファ226内の再構成されたビデオは、表示デバイスを駆動するために送信されるだけでなく、将来のビデオブロックを予測するために使用されることもある。ループ内フィルタ228がオンになっている状況では、これらの再構成されたピクセルに対してフィルタリング操作が実行され、最終的な再構成されたビデオ出力232が導出される。

【0035】

【0069】図2は、ブロックベースのビデオデコーダの一般的なブロック図を示す。ビデオビットストリームは、まずエントロピー復号ユニットでエントロピー復号される。符号化モードと予測情報は、空間予測ユニット（イントラ符号化の場合）または時間予測ユニット（インター符号化の場合）に送信されて、予測ブロックが形成される。残差変換係数は逆量子化ユニットおよび逆変換ユニットに送信され、残差ブロックが再構成される。次に、予測ブロックと残差ブロックが加算される。再構成されたブロックは、参照ピクチャストアに記憶される前に、さらにループ内フィルタリングを通過し得る。参照ピクチャストア内の再構成されたビデオは、表示デバイスを駆動するために送信されるだけでなく、将来のビデオブロックを予測するために使用される。

【0036】

【0070】一般に、V V Cに適用される基本的なイントラ予測方式は、いくつかのモジュールがさらに拡張および／または改良されている点を除いて、H E V Cのそれと同じに保たれ、例えば、イントラサブ分割（I S P）符号化モード、広角イントラ方向による拡張イントラ予測、位置依存イントラ予測組み合わせ（P D P C）、および4タップイントラ内挿などである。

10

20

30

40

50

【0037】

[0071]<VVCにおけるピクチャ、タイルグループ、タイル、およびCTUの分割>

【0038】

[0072]VVCでは、タイルは、ピクチャ内の特定のタイル列および特定のタイル行内のCTUの長方形領域として定義される。タイルグループは、単一のNALユニットに排他的に含まれるピクチャの整数のタイルのグループである。基本的に、タイルグループの概念は、HEVCで定義されているスライスと同じである。例えば、ピクチャはタイルグループとタイルに分割される。タイルは、ピクチャの長方形の領域をカバーする一連のCTUである。タイルグループには、ピクチャの多数のタイルが含まれる。タイルグループの2つのモード、つまりラスタスキャンタイルグループモードと長方形タイルグループモードがサポートされている。ラスタスキャンタイルグループモードでは、タイルグループには、ピクチャのタイルラスタスキャン内の一連のタイルが含まれる。長方形タイルグループモードでは、タイルグループには、集合的にピクチャの長方形領域を形成するピクチャの多数のタイルが含まれる。長方形タイルグループ内のタイルは、タイルグループのタイルラスタスキャンの順序になる。

【0039】

[0073]図4は、ピクチャのラスタスキャンタイルグループ分割の一例を示しており、ピクチャは12個のタイルと3個のラスタスキャンタイルグループに分割されている。図4には、タイル410、412、414、416、および418が含まれる。各タイルには18個のCTUがある。より具体的には、図4は、12個のタイルと3個のタイルグループに分割された18×12輝度CTUを含むピクチャを示している（参考）。3つのタイルグループは次のとおりである。（1）第1のタイルグループはタイル410と412を含み、（2）第2のタイルグループはタイル414、416、418、420、および422を含み、（3）第3のタイルグループはタイル424、426、428、430、432を含む。

【0040】

[0074]図5は、ピクチャを長方形タイルグループに分割する例を示している。ピクチャは、24個のタイル（タイル列6個、タイル行4個）と9個の長方形タイルグループに分割されている。図5には、タイル510、512、514、516、518、520、522、524、526、528、530、532、534、536、538、540、542、544、546、548、550、552、554、および556が含まれる。より具体的には、図5は、24個のタイルと9個のタイルグループに分割された18×12輝度CTUを有するピクチャを示す（参考）。タイルグループにはタイルが含まれ、タイルにはCTUが含まれる。9つの長方形タイルグループには、（1）2つのタイル510および512、（2）2つのタイル514および516、（3）2つのタイル518および520、（4）4つのタイル522、524、534、および536、（5）4つのタイルグループ526、528、538、540、（6）4つのタイル530、532、542、544、（7）2つのタイル546、548、（8）2つのタイル550、552、（9）2つのタイル554および556が含まれる。

【0041】

[0075]<VVCにおける高周波数ゼロアウトを伴う大きなブロックサイズの変換>

【0042】

[0076]VTM4では、最大64×64サイズの大きなブロックサイズの変換が可能であり、これは主に、より高解像度のビデオ、例えば1080pおよび4Kシーケンスに有用である。サイズ（幅もしくは高さ、または幅と高さの両方）が64に等しい変換ブロックの高周波変換係数はゼロに設定されるため、低周波係数のみが保持される。例えば、ブロック幅がM、ブロック高さがNであるM×N変換ブロックの場合、Mが64に等しい場合、変換係数の左32列のみが保持される。同様に、Nが64に等しい場合、変換係数の上位32行のみが保持される。変換スキップモードが大きなブロックに使用される場合、値をゼロにすることなくブロック全体が使用される。

【0043】

[0077]<VVCにおける仮想パイプラインデータユニット(VPDU)>

【0044】

[0078]仮想パイプラインデータユニット(VPDU)は、ピクチャ内の非重複ユニットとして定義される。ハードウェアデコーダでは、連続するVPDUが複数のパイプラインステージによって同時に処理される。VPDUサイズは、ほとんどのパイプラインステージのバッファサイズにほぼ比例するため、VPDUサイズを小さく保つことが重要である。ほとんどのハードウェアデコーダでは、VPDUサイズを最大変換ブロック(TB)サイズに設定できる。しかし、VVCでは、3値ツリー(TT)と2値ツリー(BT)の分割によってVPDUサイズが増加し得る。

10

【0045】

[0079]VPDUサイズを 64×64 輝度(luma)サンプルとして維持するために、(シンタックスの伝送の変更を伴う)以下の規範的な分割制限がVTM5に適用される。

【0046】

[0080]TT分割は、幅もしくは高さのいずれか、または幅と高さの両方が128に等しいCUでは許可されない。

【0047】

[0081] $N \leq 64$ (すなわち、幅が128に等しく、高さが128より小さい)の $128 \times N$ CUの場合、水平BTは許可されない。

【0048】

[0082] $N \leq 64$ の $N \times 128$ CU (すなわち、高さが128に等しく、幅が128より小さい)の場合、垂直BTは許可されない。

20

【0049】

[0083]図6A、6B、6C、6D、6E、6F、6G、および6Hは、VTMにおいて許可されないTTおよびBT分割の例を示す。

【0050】

[0084]<VVCにおける変換係数コーディング>

【0051】

[0085]VVCにおける変換係数コーディングは、両方とも非重複係数グループ(CGまたはサブブロックとも呼ばれる)を使用するという意味でHEVCに類似している。しかし、それらの間にはいくつかの違いもある。HEVCでは、係数の各CGのサイズは 4×4 に固定されている。VVCドラフト6では、CGサイズがTBサイズに依存するようになる。その結果、VVCではさまざまなCGサイズ(1×16 、 2×8 、 8×2 、 2×4 、 4×2 、および 16×1)が使用可能になる。符号化ブロック内のCGとCG内の変換係数は、事前に定義されたスキャン順序に従って符号化される。

30

【0052】

[0086]ピクセルあたりのコンテキスト符号化ビンの最大数を制限するには、TBの面積とビデオ成分のタイプ(例えば、輝度(luma)成分と色差(chroma)成分)を使用して、TBのコンテキスト符号化ビン(CCB)の最大数が導出される。コンテキスト符号化されたビンの最大数は、 $TB_z o s i z e * 1.75$ に等しくなる。ここで、 $TB_z o s i z e$ は、係数ゼロアウト後のTB内のサンプル数を示す。なお、 $c o d e d_s u b_b l o c k_f l a g$ は、CGに非ゼロ係数が含まれているか否かを示すフラグであり、CCBカウントには考慮されない。

40

【0053】

[0087]係数ゼロアウトは、変換ブロックの特定の領域に位置する係数を強制的に0にするために変換ブロックに対して実行される演算である。例えば、現在のVVCでは、 64×64 変換にゼロアウト演算が関連付けられている。その結果、 64×64 変換ブロック内の左上の 32×32 領域の外側に位置する変換係数はすべて強制的に0になる。実際、現在のVVCでは、特定の次元に沿ったサイズが32を超える変換ブロックについては、その次元に沿って係数ゼロアウト演算が実行され、左上の 32×32 領域を超えて位置す

50

る係数が強制的に 0 になる。

【0054】

[0088] VVC における変換係数コーディングでは、変数 `remBinsPass1` が、最初に、許容されるコンテキスト符号化ビン (MCCB) の最大数に設定される。符号化プロセスでは、コンテキストで符号化されたビンが伝送されるたびに変数が 1 ずつ減る。`remBinsPass1` が 4 以上である間、係数は最初に `sig__coeff__flag`、`abs__level__gt1__flag`、`par__level__flag`、および `abs__level__gt3__flag` のシンタックスを通じて伝送され、すべて第 1 のパスでコンテキスト符号化されたビンを使用する。係数のレベル情報の残りの部分は、第 2 のパスでゴロムライスコードとバイパス符号化されたビンを使用して、`abs__remainder` のシンタックス要素で符号化される。第 1 のパスの符号化中に `remBinsPass1` が 4 より小さくなった場合、現在の係数は第 1 のパスでは符号化されないが、ゴロムライスコードとバイパス符号化されたビンを使用して `dec__abs__level` のシンタックス要素で第 2 のパスで直接符号化される。`dec__abs__level[]` のライスパラメータ導出プロセスは、テーブル 3 に指定されているように導出される。上記のすべてのレベル符号化の後、`sig__coeff__flag` が 1 に等しいすべてのスキャン位置の符号 (符号フラグ) が最終的にバイパスビンとして符号化される。このようなプロセスを図 7 に示す。`remBinsPass1` は TB ごとにリセットされる。`sig__coeff__flag`、`abs__level__gt1__flag`、`par__level__flag`、および `abs__level__gt3__flag` のコンテキスト符号化されたビンの使用から、残りの係数のバイパス符号化されたビンの使用への移行は、TB ごとに最大 1 回のみ発生する。係数サブブロックの場合、第 1 の係数を符号化する前に `remBinsPass1` が 4 より小さい場合、係数サブブロック全体がバイパス符号化されたビンを使用して符号化される。

【0055】

[0089] 図 7 は、変換ブロックの残差符号化構造を示す。

【0056】

[0090] 統一された (同じ) ライスパラメータ (`RicePara`) 導出は、`abs__remainder` および `dec__abs__level` のシンタックスを伝送するために使用される。唯一の違いは、ベースレベル (`baseLevel`) が、`abs__remainder` と `dec__abs__level` の符号化に対してそれぞれ 4 と 0 に設定されていることである。ライスパラメータは、ローカルテンプレート内の隣接する 5 つの変換係数の絶対レベルの合計だけでなく、次のように対応するベースレベルにも基づいて決定される。

【0057】

[0091] `RicePara = RiceParTable [max (min (31, sum Abs - 5 * baseLevel), 0)]`

【0058】

[0092] 現在の VVC ドラフト仕様における残差コーディングのシンタックスおよび関連するセマンティクス (意味内容) を、それぞれテーブル 1 およびテーブル 2 に示す。テーブル 1 の読み方は、本開示の付録セクションに示されており、VVC 仕様にも記載されている。

【0059】

10

20

30

40

50

【表 1 - 1】

テーブル1. 残差コーディングのシンタックス

residual_coding(x0, y0, log2TbWidth, log2TbHeight, cIdx) {	記述子
if(sps_mts_enabled_flag && cu_sbt_flag && cIdx == 0 && log2TbWidth == 5 && log2TbHeight < 6)	
log2ZoTbWidth = 4	
else	
log2ZoTbWidth = Min(log2TbWidth, 5)	
if(sps_mts_enabled_flag && cu_sbt_flag && cIdx == 0 && log2TbWidth < 6 && log2TbHeight == 5)	
log2ZoTbHeight = 4	
else	
log2ZoTbHeight = Min(log2TbHeight, 5)	
if(log2TbWidth > 0)	
last_sig_coeff_x_prefix	ae(v)
if(log2TbHeight > 0)	
last_sig_coeff_y_prefix	ae(v)
if(last_sig_coeff_x_prefix > 3)	
last_sig_coeff_x_suffix	ae(v)
if(last_sig_coeff_y_prefix > 3)	
last_sig_coeff_y_suffix	ae(v)
log2TbWidth = log2ZoTbWidth	
log2TbHeight = log2ZoTbHeight	
remBinsPass1 = ((1 << (log2TbWidth + log2TbHeight)) * 7) >> 2	
log2SbW = (Min(log2TbWidth, log2TbHeight) < 2 ? 1 : 2)	
log2SbH = log2SbW	
if(log2TbWidth + log2TbHeight > 3)	

10

20

30

40

50

【表 1 - 2】

if(log2TbWidth < 2) {	
log2SbW = log2TbWidth	
log2SbH = 4 - log2SbW	
} else if(log2TbHeight < 2) {	
log2SbH = log2TbHeight	
log2SbW = 4 - log2SbH	
}	
numSbCoeff = 1 << (log2SbW + log2SbH)	
lastScanPos = numSbCoeff	
lastSubBlock =	
(1 << (log2TbWidth + log2TbHeight - (log2SbW + log2SbH))) - 1	
do {	
if(lastScanPos == 0) {	
lastScanPos = numSbCoeff	
lastSubBlock--	
}	
lastScanPos--	
xS =	
DiagScanOrder[log2TbWidth - log2SbW][log2TbHeight - log2SbH]	
[lastSubBlock][0]	
yS =	
DiagScanOrder[log2TbWidth - log2SbW][log2TbHeight - log2SbH]	
[lastSubBlock][1]	
xC = (xS << log2SbW) +	
DiagScanOrder[log2SbW][log2SbH][lastScanPos][0]	
yC = (yS << log2SbH) +	
DiagScanOrder[log2SbW][log2SbH][lastScanPos][1]	

10

20

30

40

50

【表 1 - 3】

<pre> } while((xC != LastSignificantCoeffX) (yC != LastSignificantCoeffY)) </pre>	
<pre> if(lastSubBlock == 0 && log2TbWidth >= 2 && log2TbHeight >= 2 && !transform_skip_flag[x0][y0][cIdx] && lastScanPos > 0) </pre>	
LfstDcOnly = 0	
<pre> if((lastSubBlock > 0 && log2TbWidth >= 2 && log2TbHeight >= 2) (lastScanPos > 7 && (log2TbWidth == 2 log2TbWidth == 3) && log2TbWidth == log2TbHeight)) </pre>	
LfstZeroOutSigCoeffFlag = 0	
if((lastSubBlock > 0 lastScanPos > 0) && cIdx == 0)	
MtsDcOnly = 0	
QState = 0	
for(i = lastSubBlock; i >= 0; i--) {	
startQStateSb = QState	
<pre> xS = DiagScanOrder[log2TbWidth - log2SbW][log2TbHeight - log2SbH] [i][0] </pre>	
<pre> yS = DiagScanOrder[log2TbWidth - log2SbW][log2TbHeight - log2SbH] [i][1] </pre>	
inferSbDcSigCoeffFlag = 0	
if(i < lastSubBlock && i > 0) {	
sb_coded_flag [xS][yS]	ae(v)
inferSbDcSigCoeffFlag = 1	
}	
if(sb_coded_flag[xS][yS] && (xS > 3 yS > 3) && cIdx == 0)	

10

20

30

40

50

【表 1 - 4】

MtsZeroOutSigCoeffFlag = 0	
firstSigScanPosSb = numSbCoeff	
lastSigScanPosSb = -1	
firstPosMode0 = (i == lastSubBlock ? lastScanPos : numSbCoeff - 1)	
firstPosMode1 = firstPosMode0	
for(n = firstPosMode0; n >= 0 && remBinsPass1 >= 4; n--) {	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
if(sb_coded_flag[xS][yS] && (n > 0 !inferSbDcSigCoeffFlag) && (xC != LastSignificantCoeffX yC != Last SignificantCoeffY)) {	
sig_coeff_flag [xC][yC]	ae(v)
remBinsPass1--	
if(sig_coeff_flag[xC][yC])	
inferSbDcSigCoeffFlag = 0	
}	
if(sig_coeff_flag[xC][yC]) {	
abs_level_gtx_flag [n][0]	ae(v)
remBinsPass1--	
if(abs_level_gtx_flag[n][0]) {	
par_level_flag [n]	ae(v)
remBinsPass1--	
abs_level_gtx_flag [n][1]	ae(v)
remBinsPass1--	
}	

10

20

30

40

50

【表 1 - 5】

if(lastSigScanPosSb == -1)	
lastSigScanPosSb = n	
firstSigScanPosSb = n	
}	
AbsLevelPass1[xC][yC] = sig_coeff_flag[xC][yC] + par_level_flag[n] + abs_level_gtx_flag[n][0] + 2 * abs_level_gtx_flag[n][1]	10
if(sh_dep_quant_used_flag)	
QState = QStateTransTable[QState][AbsLevelPass1[xC][yC] & 1]	
firstPosMode1 = n - 1	
}	
for(n = firstPosMode0; n > firstPosMode1; n--) {	20
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
if(abs_level_gtx_flag[n][1])	
abs_remainder [n]	ae(v)
AbsLevel[xC][yC] = AbsLevelPass1[xC][yC] + 2 * abs_remainder[n]	30
}	
for(n = firstPosMode1; n >= 0; n--) {	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	40

10

20

30

40

50

【表 1 - 6】

if(sb_coded_flag[xS][yS])	
dec_abs_level[n]	ae(v)
if(AbsLevel[xC][yC] > 0) {	
if(lastSigScanPosSb == -1)	
lastSigScanPosSb = n	
firstSigScanPosSb = n	
}	
if(sh_dep_quant_used_flag)	
QState = QStateTransTable[QState][AbsLevel[xC][yC] & 1]	
}	
signHiddenFlag = sh_sign_data_hiding_used_flag && (lastSigScanPosSb - firstSigScanPosSb > 3 ? 1 : 0)	
for(n = numSbCoeff - 1; n >= 0; n--) {	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
if((AbsLevel[xC][yC] > 0) && (!signHiddenFlag (n != firstSigScanPosSb)))	
coeff_sign_flag[n]	ae(v)
}	
if(sh_dep_quant_used_flag) {	
QState = startQStateSb	
for(n = numSbCoeff - 1; n >= 0; n--) {	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	

10

20

30

40

50

【表 1 - 7】

if(AbsLevel[xC][yC] > 0)	
TransCoeffLevel[x0][y0][cIdx][xC][yC] = (2 * AbsLevel[xC][yC] - (QState > 1 ? 1 : 0)) * (1 - 2 * coeff_sign_flag[n])	
QState = QStateTransTable[QState][AbsLevel[xC][yC] & 1]	
} else {	
sumAbsLevel = 0	
for(n = numSbCoeff - 1; n >= 0; n--) {	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
if(AbsLevel[xC][yC] > 0) {	
TransCoeffLevel[x0][y0][cIdx][xC][yC] = AbsLevel[xC][yC] * (1 - 2 * coeff_sign_flag[n])	
if(signHiddenFlag) {	
sumAbsLevel += AbsLevel[xC][yC]	
if((n == firstSigScanPosSb) && (sumAbsLevel % 2) == 1))	
TransCoeffLevel[x0][y0][cIdx][xC][yC] = -TransCoeffLevel[x0][y0][cIdx][xC][yC]	
}	
}	
}	
}	
}	
}	

10

20

30

40

【 0 0 6 0 】

50

【表 2 - 1】

テーブル2. 残差コーディングのセマンティクス

配列AbsLevel[xC][yC]は、現在の変換ブロックの変換係数レベルの絶対値の配列を表す。配列AbsLevelPass1[xC][yC]は、現在の変換ブロックの変換係数レベルの部分的に再構成された絶対値の配列を表す。配列インデックスxCおよびyCは、現在の変換ブロック内の変換係数の位置(xC, yC)を指定する。AbsLevel[xC][yC]の値が7. 3. 11. 11節で指定されていない場合、0に等しいと推定される。AbsLevelPass1[xC][yC]の値が7. 3. 11. 11節で指定されていない場合、0に等しいと推定される。

10

最小および最大の変換係数値を指定する変数CoeffMinおよびCoeffMaxは、次のように導出される。

$$\text{CoeffMin} = -(1 \ll 15) \quad (189)$$

$$\text{CoeffMax} = (1 \ll 15) - 1 \quad (190)$$

配列QStateTransTable[][]は次のように指定される。

20

$$\text{QStateTransTable}[][] = \{ \{ 0, 2 \}, \{ 2, 0 \}, \{ 1, 3 \}, \{ 3, 1 \} \} \quad (191)$$

last_sig_coeff_x_prefixは、変換ブロック内のスキャン順序で最後の有効な係数の列位置のプレフィックスを指定する。last_sig_coeff_x_prefixの値は、 $0 \sim (\log_2 \text{ZoTbWidth} \ll 1) - 1$ の範囲内でなければならない(両端を含む)。

last_sig_coeff_x_prefixが存在しない場合、0であると推定される。

last_sig_coeff_y_prefixは、変換ブロック内のスキャン順序で最後の有効な係数の行位置のプレフィックスを指定する。last_sig_coeff_y_prefixの値は、 $0 \sim (\log_2 \text{ZoTbHeight} \ll 1) - 1$ の範囲内でなければならない(両端を含む)。

30

last_sig_coeff_y_prefixが存在しない場合、0であると推定される。

last_sig_coeff_x_suffixは、変換ブロック内のスキャン順序で最後の有効な係数の列位置のサフィックスを指定する。last_sig_coeff_x_suffixの値は、 $0 \sim$

$(1 \ll ((\text{last_sig_coeff_x_prefix} \gg 1) - 1)) - 1$ の範囲内でなければならない(両端を含む)。

40

【表 2 - 2】

変換ブロック LastSignificantCoeffX 内のスキャン順序における最後の有効係数の列位置は、次のように導出される。

- last_sig_coeff_x_suffixが存在しない場合は、以下が適用される。

$$\text{LastSignificantCoeffX} = \text{last_sig_coeff_x_prefix} \quad (192)$$

- それ以外の場合 (last_sig_coeff_x_suffix が存在する場合)、以下が適用される。

$$\text{LastSignificantCoeffX} = (1 \ll ((\text{last_sig_coeff_x_prefix} \gg 1) - 1)) * (2 + (\text{last_sig_coeff_x_prefix} \& 1)) + \text{last_sig_coeff_x_suffix} \quad (193)$$

last_sig_coeff_y_suffix は、変換ブロック内のスキャン順序で最後の有効な係数の行位置のサフィックスを指定する。last_sig_coeff_y_suffix の値は、0～

(1 << ((last_sig_coeff_y_prefix >> 1) - 1)) - 1 までの範囲でなければならない(両端を含む)。

変換ブロック LastSignificantCoeffY 内のスキャン順序における最後の有効係数の行位置は、次のように導出される。

- last_sig_coeff_y_suffix が存在しない場合は、以下が適用される。

$$\text{LastSignificantCoeffY} = \text{last_sig_coeff_y_prefix} \quad (194)$$

- それ以外の場合 (last_sig_coeff_y_suffix が存在する場合)、以下が適用される。

$$\text{LastSignificantCoeffY} = (1 \ll ((\text{last_sig_coeff_y_prefix} \gg 1) - 1)) * (2 + (\text{last_sig_coeff_y_prefix} \& 1)) + \text{last_sig_coeff_y_suffix} \quad (195)$$

sb_coded_flag[xS][yS]は、現在の変換ブロック内の位置(xS, yS)にあるサブブロックに対して以下を指定する。ここで、サブブロックは変換係数レベルの配列である。

sb_coded_flag[xS][yS]が0に等しい場合、位置(xS, yS)にあるサブブロックのすべての変換係数レベルは0に等しいと推定される。

sb_coded_flag[xS][yS]が存在しない場合、1に等しいと推定される。

sig_coeff_flag[xC][yC]は、現在の変換ブロック内の変換係数の位置(xC, yC)について、位置(xC, yC)での対応する変換係数レベルがゼロ以外であるかどうかを次のように指定する。

- sig_coeff_flag[xC][yC]が0に等しい場合、位置(xC, yC)での変換係数レベルは0に設定される。

- それ以外の場合 (sig_coeff_flag[xC][yC]が1に等しい)、位置(xC, yC)の変換係数レベルはゼロ以外の値になる。

10

20

30

40

50

【表 2 - 3】

sig_coeff_flag[xC][yC]が存在しない場合、次のように推定される。

— transform_skip_flag[x0][y0][cIdx]が0に等しい場合、または
sh_ts_residual_coding_disabled_flag が1に等しい場合、以下が適用される。

— (xC, yC) がスキャン順で最後の重要な位置 (LastSignificantCoeffX, LastSignificantCoeffY) である場合、または以下の条件がすべて真の場合、
sig_coeff_flag[xC][yC]は1と等しいと推定される。

— (xC & ((1 << log2SbW) - 1), yC & ((1 << log2SbH) - 1)) は (0, 0) に等しい。

— inferSbDcSigCoeffFlag は1に等しい。

— sb_coded_flag[xS][yS]は1に等しい。

— それ以外の場合、sig_coeff_flag[xC][yC]は0に等しいと推定される。

— それ以外の場合 (transform_skip_flag[x0][y0][cIdx]が1に等しく、
sh_ts_residual_coding_disabled_flag が0に等しい)、以下が適用される。

— 次の条件がすべて真の場合、sig_coeff_flag[xC][yC]は1に等しいと推定される。

— (xC & ((1 << log2SbW) - 1), yC & ((1 << log2SbH) - 1)) は、((1 << log2SbW) ? 1, (1 << log2SbH) ? 1) に等しい。

— inferSbSigCoeffFlag は1に等しい。

— sb_coded_flag[xS][yS]は1に等しい。

— それ以外の場合、sig_coeff_flag[xC][yC]は0に等しいと推定される。

abs_level_gtx_flag[n][j]は、(スキャン位置nでの)変換係数レベルの絶対値が(j << 1) + 1より大きいかどうかを指定する。abs_level_gtx_flag[n][j]が存在しない場合は、0に等しいと推定される。

par_level_flag[n]は、スキャン位置nにおける変換係数レベルのパリティを指定する。
par_level_flag[n]が存在しない場合、0に等しいと推定される。

abs_remainder[n]は、スキャン位置nにおけるゴロムライス符号で符号化された変換係数レベルの残りの絶対値である。abs_remainder[n]が存在しない場合、0に等しいと推定される。

abs_remainder[n]の値は、TransCoeffLevel[x0][y0][cIdx][xC][yC]の対応する値がCoeff MinからCoeffMaxまでの範囲内(両端を含む)になるように制約されるものとすることがビットストリーム準拠の要件である。

10

20

30

40

50

【表 2 - 4】

dec_abs_level[n]は、スキャン位置 n におけるゴロムライス符号で符号化された中間値である。**dec_abs_level[n]**の解析中にテーブル3で導出される **ZeroPos[n]**を指定すると、位置(x_C , y_C)での変換係数レベルの絶対値 **AbsLevel[xC][yC]**が次のように導出される。

- **dec_abs_level[n]**が存在しないか、**ZeroPos[n]**に等しい場合、**AbsLevel[xC][yC]**は0に設定される。
- それ以外の場合、**dec_abs_level[n]**が **ZeroPos[n]**より小さい場合、**AbsLevel[xC][yC]**は **dec_abs_level[n] + 1** に等しく設定される。
- それ以外の場合 (**dec_abs_level[n]**が **ZeroPos[n]**より大きい)、**AbsLevel[xC][yC]**は **dec_abs_level[n]**に等しく設定される。

dec_abs_level[n]の値が、**TransCoeffLevel[x0][y0][cIdx][xC][yC]**の対応する値が **CoeffMin**から **Coeff**の範囲内(両端を含む)になるように制約されるものとするは、ビットストリーム準拠の要件である。

coeff_sign_flag[n]は、スキャン位置 n の変換係数レベルの符号を次のように指定する。

- **coeff_sign_flag[n]**が0に等しい場合、対応する変換係数レベルは正の値になる。
- それ以外の場合 (**coeff_sign_flag[n]**が1に等しい)、対応する変換係数レベルは負の値になる。

coeff_sign_flag[n]が存在しない場合、0に等しいと推定される。

CoeffSignLevel[xC][yC]の値は、位置(x_C , y_C)での変換係数レベルの符号を次のように指定する。

- **CoeffSignLevel[xC][yC]**が0に等しい場合、対応する変換係数レベルは0に等しい
- それ以外の場合、**CoeffSignLevel[xC][yC]**が1に等しい場合、対応する変換係数レベルは正の値になる。
- それ以外の場合 (**CoeffSignLevel[xC][yC]**が-1に等しい)、対応する変換係数レベルは負の値になる。

【 0 0 6 1 】

10

20

30

40

50

【表 3】

テーブル3. abs_remainder[]およびdec_abs_level[]のライスパラメータ導出プロセス

このプロセスへの入力は、ベースレベルのbaseLevel、カラー成分インデックスcldx、現在のピクチャの左上のサンプルを基準とした現在の変換ブロックの左上のサンプルを指定する輝度位置(x0, y0)、現在の係数スキャン位置(xC, yC)、変換ブロック幅の2進対数log2TbWidth、および変換ブロックの高さの2進対数log2TbHeightである。

このプロセスの出力は、ライスパラメータcRiceParamである。

成分インデックスcldxと左上の輝度位置(x0, y0)を持つ変換ブロックの配列AbsLevel[x][y]を指定すると、変数locSumAbsは次の疑似コードで指定されているように導出される。

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

変数locSumAbsを指定すると、ライスパラメータcRiceParamは以下のテーブル4に示されるように導出される。

baseLevelが0に等しい場合、変数ZeroPos[n]は次のように導出される。

$ZeroPos[n] = (QState < 2 ? 1 : 2) << cRiceParam$

【 0 0 6 2 】

10

20

30

40

50

【表 4】

テーブル4. locSumAbsに基づくcRiceParamの仕様

locSumAbs	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
cRiceParam	0	0	0	0	0	0	0	1	1	1	1	1	1	1	2	2
locSumAbs	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
cRiceParam	2	2	2	2	2	2	2	2	2	2	2	2	3	3	3	3

10

【0063】

[0093] < V V C における変換スキップモードの残差コーディング >

【0064】

[0094] 単一の残差コーディング方式が変換係数と変換スキップ係数の両方を符号化するために設計される H E V C とは異なり、V V C では、2つの別個の残差コーディング方式がそれぞれ変換係数および変換スキップ係数（すなわち、残差）に使用される。

【0065】

[0095] 変換スキップモードでは、残差信号の統計的特性は変換係数の統計的特性とは異なり、低周波成分の周囲のエネルギー圧縮は観察されない。残差コーディングは、次のような（空間）変換スキップ残差のさまざまな信号特性を考慮して変更される。

20

【0066】

[0096] 最後の x / y 位置の伝送がない。

【0067】

[0097] 前のすべてのフラグが 0 に等しい場合、D C サブブロックを除くすべてのサブブロックに対して符号化された `coded_sub_block_flag`。

【0068】

[0098] 2つの隣接する係数を使用した `sig_coeff_flag` コンテキストモデリング。

【0069】

[0099] 1つのコンテキストモデルのみを使用する `par_level_flag`。

30

【0070】

[00100] 5、7、9を超える追加のフラグ。

【0071】

[00101] 剰余 2 値化のための修正されたライスパラメータ導出。

【0072】

[00102] 符号フラグのコンテキストモデリングは、左および上の隣接する係数値に基づいて決定され、符号フラグは、すべてのコンテキスト符号化ピンをまとめて維持するために、`sig_coeff_flag` の後に解析される。

【0073】

[00103] 図 8 に示すように、シンタックス要素 `sig_coeff_flag`、`coeff_sign_flag`、`abs_level_gt1_flag`、`par_level_flag` は、第 1 のパスで残差サンプルごとにインターリーブ方式で符号化され、続いて第 2 のパスで `abs_level_gtX_flag` ビットプレーンが続き、第 3 のパスで `abs_remainder` 符号化が続く。

40

【0074】

[00104] パス 1 : `sig_coeff_flag`、`coeff_sign_flag`、`abs_level_gt1_flag`、`par_level_flag`

【0075】

[00105] パス 2 : `abs_level_gt3_flag`、`abs_level_gt`

50

5__flag、abs__level__gt7__flag、abs__level__gt9__flag

【0076】

[00106]パス3：abs__remainder

【0077】

[00107]図8は、変換スキップブロックの残差符号化構造を示す。

【0078】

[00108]現在のVVCドラフト仕様における変換スキップモードの残差コーディングのシンタックスおよび関連するセマンティクス（意味内容）を、それぞれテーブル5およびテーブル2に示す。テーブル5の読み方は、本開示の付録セクションに示されており、VVC仕様にも記載されている。

【0079】

10

20

30

40

50

【表 5 - 1】

テーブル5. 変換スキップモードの残差コーディングのシンタックス

residual_ts_coding(x0, y0, log2TbWidth, log2TbHeight, cIdx) {	記述子
log2SbW = (Min(log2TbWidth, log2TbHeight) < 2 ? 1 : 2)	
log2SbH = log2SbW	
if(log2TbWidth + log2TbHeight > 3)	
if(log2TbWidth < 2) {	
log2SbW = log2TbWidth	
log2SbH = 4 - log2SbW	
} else if(log2TbHeight < 2) {	
log2SbH = log2TbHeight	
log2SbW = 4 - log2SbH	
}	
numSbCoeff = 1 << (log2SbW + log2SbH)	
lastSubBlock =	
(1 << (log2TbWidth + log2TbHeight - (log2SbW + log2SbH))) - 1	
inferSbCbf = 1	
RemCbs = ((1 << (log2TbWidth + log2TbHeight)) * 7) >> 2	
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 !inferSbCbf)	
sb_coded_flag [xS][yS]	ae(v)
if(sb_coded_flag[xS][yS] && i < lastSubBlock)	
inferSbCbf = 0	

10

20

30

40

50

【表 5 - 2】

/* First scan pass */	
inferSbSigCoeffFlag = 1	
lastScanPosPass1 = -1	
for(n = 0; n <= numSbCoeff - 1 && RemCpbs >= 4; n++) {	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
lastScanPosPass1 = n	
if(sb_coded_flag[xS][yS] && (n != numSbCoeff - 1 !inferSbSigCoeffFlag)) {	
sig_coeff_flag [xC][yC]	ae(v)
RemCpbs--	
if(sig_coeff_flag[xC][yC])	
inferSbSigCoeffFlag = 0	
}	
CoeffSignLevel[xC][yC] = 0	
if(sig_coeff_flag[xC][yC]) {	
coeff_sign_flag [n]	ae(v)
RemCpbs--	
CoeffSignLevel[xC][yC] = (coeff_sign_flag[n] > 0 ? -1 : 1)	
abs_level_gtx_flag [n][0]	ae(v)
RemCpbs--	
if(abs_level_gtx_flag[n][0]) {	
par_level_flag [n]	ae(v)
RemCpbs--	
}	
}	

10

20

30

40

50

【表 5 - 3】

AbsLevelPass1[xC][yC] = sig_coeff_flag[xC][yC] + par_level_flag[n] + abs_level_gtx_flag[n][0]	
}	
/* Greater than X scan pass (numGtXFlags=5) */	
lastScanPosPass2 = -1	
for(n = 0; n <= numSbCoeff - 1 && RemCpbs >= 4; n++) {	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	
AbsLevelPass2[xC][yC] = AbsLevelPass1[xC][yC]	
for(j = 1; j < 5; j++) {	
if(abs_level_gtx_flag[n][j - 1]) {	
abs_level_gtx_flag[n][j]	ae(v)
RemCpbs--	
}	
AbsLevelPass2[xC][yC] += 2 * abs_level_gtx_flag[n][j]	
}	
lastScanPosPass2 = n	
}	
/* remainder scan pass */	
for(n = 0; n <= numSbCoeff - 1; n++) {	
xC = (xS << log2SbW) + DiagScanOrder[log2SbW][log2SbH][n][0]	
yC = (yS << log2SbH) + DiagScanOrder[log2SbW][log2SbH][n][1]	

10

20

30

40

50

【表 5 - 4】

<pre> if((n <= lastScanPosPass2 && AbsLevelPass2[xC][yC] >= 10) (n > lastScanPosPass2 && n <= lastScanPosPass1 && AbsLevelPass1[xC][yC] >= 2) (n > lastScanPosPass1 && sb_coded_flag[xS][yS])) </pre>		
abs_remainder[n]	ae(v)	
if(n <= lastScanPosPass2)		10
<pre> AbsLevel[xC][yC] = AbsLevelPass2[xC][yC] + 2 * abs_remainder[n] </pre>		
else if(n <= lastScanPosPass1)		
<pre> AbsLevel[xC][yC] = AbsLevelPass1[xC][yC] + 2 * abs_remainder[n] </pre>		
else { /* bypass */		
AbsLevel[xC][yC] = abs_remainder[n]		20
if(abs_remainder[n])		
coeff_sign_flag[n]	ae(v)	
}		
if(BdpcmFlag[x0][y0][cIdx] == 0 && n <= lastScanPosPass1) {		
absLeftCoeff = xC > 0 ? AbsLevel[xC - 1][yC] : 0		
absAboveCoeff = yC > 0 ? AbsLevel[xC][yC - 1] : 0		30
predCoeff = Max(absLeftCoeff, absAboveCoeff)		
if(AbsLevel[xC][yC] == 1 && predCoeff > 0)		
AbsLevel[xC][yC] = predCoeff		
else if(AbsLevel[xC][yC] > 0 && AbsLevel[xC][yC] <= predCoeff)		
AbsLevel[xC][yC] --		
}		40

10

20

30

40

50

【表 5 - 5】

TransCoeffLevel[x0][y0][cIdx][xC][yC] = (1 - 2 * coeff_sign_flag[n]) * AbsLevel[xC][yC]	
}	
}	
}	

10

【 0 0 8 0 】

[00109] <量子化>

【 0 0 8 1 】

[00110]現在のVVCでは、最大QP値が51から63に拡張され、それに応じて初期QPの伝送が変更された。SliceQPYの初期値は、slice_qp_deltaの非ゼロ値が符号化されるときにスライスセグメント層で変更できる。変換スキップブロックの場合、QPが4に等しい場合、量子化ステップサイズは1になるため、最小許容量子化パラメータ（QP）は4として定義される。

20

【 0 0 8 2 】

[00111]さらに、同じHEVCスカラー量子化が、依存スカラー量子化と呼ばれる新しい概念とともに使用される。依存スカラー量子化とは、変換係数の許容可能な再構成値のセットが、再構成順序で現在の変換係数レベルより前の変換係数レベルの値に依存するアプローチを指す。このアプローチの主な効果は、HEVCで使用される従来の独立したスカラー量子化と比較して、許容される再構成ベクトルがN次元ベクトル空間でより高密度にパックされることである（Nは変換ブロック内の変換係数の数を表す）。つまり、N次元単位体積あたりの許容再構成ベクトルの平均数が一定である場合、入力ベクトルと最も近い再構成ベクトルとの間の平均歪みが減少する。依存スカラー量子化のアプローチは、（a）異なる再構成レベルを持つ2つのスカラー量子化器を定義し、（b）2つのスカラー量子化器間で切り替えるプロセスを定義することによって実現される。

30

【 0 0 8 3 】

[00112]Q0およびQ1で示される、使用される2つのスカラー量子化器が図9に示されている。利用可能な再構成レベルの位置は、量子化ステップサイズΔによって一意に指定される。使用されるスカラー量子化器（Q0またはQ1）は、ビットストリーム内で明示的に伝送されない。代わりに、現在の変換係数に使用される量子化器は、符号化／再構成の順序で現在の変換係数に先行する変換係数レベルのパリティによって決定される。

【 0 0 8 4 】

[00113]図9は、提案された依存量子化アプローチで使用される2つのスカラー量子化器の図を示す。

40

【 0 0 8 5 】

[00114]図10Aおよび10Bに示されるように、2つのスカラー量子化器（Q0およびQ1）間の切り替えは、4つの量子化器状態（QState）を有するステートマシンを介して実現される。QStateは、0、1、2、3の4つの異なる値を取ることができる。これは、符号化／再構成順序で現在の変換係数に先行する変換係数レベルのパリティによって一意に決定される。変換ブロックの逆量子化の開始時に、状態は0に設定される。変換係数は、スキャン順序で（つまり、エントロピー復号と同じ順序で）再構成される。現在の変換係数が再構成された後、状態は図10に示すように更新される。ここで、kは変換係数レベルの値を示す。

【 0 0 8 6 】

50

【00115】図10Aは、提案された依存量子化の状態遷移を示す遷移図を示す。

【0087】

【00116】図10Bは、提案された依存量子化のための量子化器の選択を示す表を示す。

【0088】

【00117】デフォルトおよびユーザ定義のスケーリング行列を伝送することもサポートされている。DEFAULTモードのスケーリング行列はすべてフラットであり、すべてのTBサイズで要素が16に相当する。IBCおよびイントラ符号化モードは現在、同じスケーリング行列を共有している。したがって、USER_DEFINED行列の場合、MatrixTypeとMatrixType_DCの数は次のように更新される。

【0089】

【00118】MatrixType: 30 = 2 (イントラ&IBC/インターの場合は2) × 3 (Y/Cb/Cr成分) × 5 (正方形のTBサイズ: 輝度 (luma) については4 × 4から64 × 64まで、色差 (chroma) については2 × 2から32 × 32まで)。

【0090】

【00119】MatrixType_DC: 14 = 2 (イントラ&IBC/インターの2 × Y成分の1) × 3 (TBサイズ: 16 × 16、32 × 32、64 × 64) + 4 (イントラ&IBC/インターの2 × Cb/Cr成分の2) × 2 (TBサイズ: 16 × 16、32 × 32)。

【0091】

【00120】DC値は、次のスケーリング行列、16 × 16、32 × 32、および64 × 64に対して個別に符号化される。8 × 8より小さいサイズのTBの場合、1つのスケーリング行列のすべての要素が伝送される。TBのサイズが8 × 8以上の場合、1つの8 × 8スケーリング行列の64要素のみがベーススケーリング行列として伝送される。8 × 8を超えるサイズの正方行列を取得するには、8 × 8のベーススケーリング行列が(要素の複製によって)対応する正方形サイズ(つまり、16 × 16、32 × 32、64 × 64)にアップサンプリングされる。64点変換の高周波係数のゼロアウトが適用されると、スケーリング行列の対応する高周波もゼロアウトされる。つまり、TBの幅または高さが32以上の場合、係数の左半分または上半分のみが保持され、残りの係数には0が割り当てられる。さらに、右下の4 × 4要素は決して使用されないため、64 × 64スケーリング行列について伝送される要素の数も8 × 8から3つの4 × 4部分行列に減る。

【0092】

【00121】<変換係数コーディングのためのコンテキストモデリング>

【0093】

【00122】変換係数レベルの絶対値に関連するシンタックス要素の確率モデルの選択は、ローカル近傍における絶対レベルまたは部分的に再構成された絶対レベルの値に依存する。使用したテンプレートを図11に示す。

【0094】

【00123】図11は、確率モデルを選択するために使用されるテンプレートの図を示す。黒い四角形は現在のスキャン位置を指定し、「x」の付いた四角形は使用されるローカル近傍を表す。

【0095】

【00124】選択される確率モデルは、ローカル近傍内の絶対レベル(または部分的に再構成された絶対レベル)の合計と、ローカル近傍内の0より大きい絶対レベルの数(1に等しいsig_coeff_flagの数によって与えられる)に依存する。コンテキストモデリングと2値化は、ローカル近傍に対する次の尺度に依存する。

【0096】

【00125】numSig: ローカル近傍内の非ゼロレベルの数。

【0097】

【00126】sumAbs1: ローカル近傍における第1のパス後の部分的に再構成された絶対レベル(absLevel1)の合計。

10

20

30

40

50

【0098】

【00127】`sumAbs`：ローカル近傍における再構成された絶対レベルの合計。

【0099】

【00128】対角位置（`d`）：変換ブロック内の現在のスキャン位置の水平座標と垂直座標の合計。

【0100】

【00129】`numSig`、`sumAbs1`、および`d`の値に基づいて、`sig__coeff__flag`、`abs__level__gt1__flag`、`par__level__flag`、および`abs__level__gt3__flag`を符号化するための確率モデルが選択される。`abs__remainder`と`dec__abs__level`を2値化するためのラ

10

イスパラメータは、`sumAbs`と`numSig`の値に基づいて選択される。

【0101】

【00130】現在のVVCでは、縮小32ポイントMTS（`RMTS32`とも呼ばれる）は、高周波係数のスキップに基づいており、32ポイントDST-7／DCT-8の計算の複雑さを軽減するために使用される。また、これには、あらゆるタイプのゼロアウト（つまり、`RMTS32`および`DCT2`の高周波成分用の既存のゼロアウト）を含む係数コーディングの変更が伴う。具体的には、最後の非ゼロ係数位置コーディングの2値化は、縮小されたTUサイズに基づいて符号化され、最後の非ゼロ係数位置コーディングのためのコンテキストモデルの選択は、元のTUサイズによって決定される。さらに、変換係数の`sig__coeff__flag`を符号化するために60のコンテキストモデルが使用される。コンテキストモデルインデックスの選択は、次のように、`locSumAbsPass1`と呼ばれる、事前に部分的に再構成された最大5つの絶対レベルの合計と、依存する量子化の状態`QState`に基づいて行われる。

20

【0102】

【00131】`cldx`が0に等しい場合、`ctxlnc`は次のように導出される：

$$\text{ctxlnc} = 12 * \text{Max}(0, \text{QState} - 1) +$$

$$\text{Min}((\text{locSumAbsPass1} + 1) >> 1, 3) +$$

$$(d < 2 ? 8 : (d < 5 ? 4 : 0))$$

【0103】

【00132】それ以外の場合（`cldx`が0より大きい）、`ctxlnc`は次のように導出される：

30

$$\text{ctxlnc} = 36 + 8 * \text{Max}(0, \text{QState} - 1) +$$

$$\text{Min}((\text{locSumAbsPass1} + 1) >> 1, 3) + (d < 2 ? 4 : 0)$$

【0104】

【00133】＜パレットモード＞

【0105】

【00134】パレットモードの背後にある基本的な考え方は、CU内のサンプルが代表的な色値の小さなセットによって表されるということである。このセットをパレットと呼ぶ。また、3つの色成分の値がビットストリームで直接伝送されるエスケープカラーとして伝送することで、パレットから除外されるカラー値を示すこともできる。これを図12に示す。

40

【0106】

【00135】図12は、パレットモードで符号化されたブロックの例を示す。図12は、パレットモードで符号化された1210のブロックと1220のパレットを含む。

【0107】

【00136】図12において、パレットサイズは4である。第1の3つのサンプルは、再構成にそれぞれパレットエントリ2、0、および3を使用する。青いサンプルはエスケープ記号を表す。CUレベルフラグ`palette__escape__val__present__flag`は、エスケープ記号がCU内に存在するかどうかを示す。エスケープ記号が存在する場合、パレットサイズは1だけ増加し、最後のインデックスがエスケープ記号を示

50

すために使用される。したがって、図 1 2 では、インデックス 4 がエスケープ記号に割り当てられている。

【0108】

[00137]パレット符号化されたブロックを復号するために、デコーダは以下の情報を有する必要がある。

[00138]パレットテーブル

[00139]パレットインデックス

【0109】

[00140]パレットインデックスがエスケープ記号に対応する場合、サンプルの対応するカラー値を示すために追加のオーバーヘッドが伝送される。

【0110】

[00141]さらに、エンコーダ側では、そのCUで使用される適切なパレットを導出する必要がある。

【0111】

[00142]非可逆符号化用のパレットを導出するために、修正されたk平均クラスタリングアルゴリズムが使用される。ブロックの第1のサンプルがパレットに追加される。次に、ブロックからの後続の各サンプルについて、サンプルと現在のパレットカラーのそれぞれの間の絶対差の合計(SAD)が計算される。各成分の歪みが最小SADに対応するパレットエントリの閾値よりも小さい場合、サンプルはそのパレットエントリに属するクラスタに追加される。それ以外の場合、サンプルは新しいパレットエントリとして追加される。クラスタにマッピングされたサンプルの数が閾値を超えると、そのクラスタの重心が更新され、そのクラスタのパレットエントリになる。

【0112】

[00143]次のステップでは、クラスタは使用量の降順にソートされる。そして、各エントリに対応するパレットエントリが更新される。通常、クラスタの重心がパレットのエントリとして使用される。しかし、パレットエントリの符号化のコストを考慮した場合、パレット予測子からのエントリが重心の代わりに更新されたパレットエントリとして使用するのに適しているかどうかを分析するために、レート歪み分析が実行される。このプロセスは、すべてのクラスタが処理されるか、パレットの最大サイズに達するまで継続される。最後に、クラスタにサンプルが1つしかなく、対応するパレットエントリがパレット予測子にない場合、サンプルはエスケープ記号に変換される。さらに、重複したパレットエントリが削除され、それらのクラスタがマージされる。

【0113】

[00144]パレット導出後、ブロック内の各サンプルには、最も近い(SAD内の)パレットエントリのインデックスが割り当てられる。次に、サンプルは「INDEX」または「COPY__ABOVE」モードに割り当てられる。「INDEX」または「COPY__ABOVE」モードのいずれかが可能なサンプルごとに。次に、モードを符号化するコストが計算される。コストの低いモードが選択される。

【0114】

[00145]パレットエントリの符号化のために、パレット予測子が維持される。パレットの最大サイズとパレット予測子は、SPSで伝送される。パレット予測子は、各CTU行、各スライス、および各タイルの先頭で初期化される。

【0115】

[00146]パレット予測子内のエントリごとに、それが現在のパレットの一部であるかどうかを示す再利用フラグが伝送される。これを図 1 3 に示す。

【0116】

[00147]図 1 3 は、パレットエントリを伝送するためのパレット予測子の使用を示す。図 1 3 は、前のパレット 1 3 1 0 と現在のパレット 1 3 2 0 を含む。

【0117】

[00148]再利用フラグは、ゼロのランレングス符号化を使用して送信される。この後、

10

20

30

40

50

次数0の指数ゴロムコードを使用して、新しいパレットエントリの数が伝送される。最後に、新しいパレットエントリの成分値が伝送される。

【0118】

[00149]パレットインデックスは、図14Aおよび14Bに示すように、水平および垂直トラバーススキャンを使用して符号化される。スキャン順序は、`palette__transpose__flag`を使用してビットストリーム内で明示的に伝送される。

【0119】

[00150]図14Aは、水平トラバーススキャンを示す。

【0120】

[00151]図14Bは、垂直トラバーススキャンを示す。

【0121】

[00152]パレットインデックスを符号化するために、ライン係数グループ(CG)ベースのパレットモードが使用され、これは、図15Aおよび15Bに示すように、トラバーススキャンモードに基づいてCUを16サンプルを有する複数のセグメントに分割し、インデックスラン、パレットインデックス値、およびエスケープモードの量子化されたカラーは、CGごとに順番に符号化／解析される。

【0122】

[00153]図15Aは、パレットをスキャンするサブブロックベースのインデックスマップを示す。

【0123】

[00154]図15Bは、パレットをスキャンするサブブロックベースのインデックスマップを示す。

【0124】

[00155]パレットインデックスは、2つの主要なパレットサンプルモード、「INDEX」および「COPY__ABOVE」を使用して符号化される。前に説明したように、エスケープ記号には最大パレットサイズに等しいインデックスが割り当てられる。「COPY__ABOVE」モードでは、上の行のサンプルのパレットインデックスがコピーされる。「INDEX」モードでは、パレットインデックスが明示的に伝送される。各セグメントのパレットラン符号化の符号化順序は次のとおりである。

【0125】

[00156]各ピクセルに対して、1つのコンテキスト符号化ピン`run__copy__flag`が伝送され、ピクセルが前のピクセルと同じモードであるかどうかを示し、つまり、以前にスキャンされたピクセルと現在のピクセルが両方ともランタイプCOPY__ABOVEである場合、または以前にスキャンされたピクセルと現在のピクセルが両方ともランタイプINDEXで同じインデックス値である場合である。それ以外の場合は、`run__copy__flag`が1が伝送される。

【0126】

[00157]ピクセルと前のピクセルが異なるモードである場合、1つのコンテキスト符号化ピン`copy__above__palette__indices__flag`が伝送され、ピクセルのランタイプ、すなわちINDEXまたはCOPY__ABOVEを示す。デフォルトでINDEXモードが使用されるため、サンプルが第1の行(水平トラバーススキャン)または第1の列(垂直トラバーススキャン)にある場合、デコーダはランタイプを解析する必要はない。また、以前に解析されたランタイプがCOPY__ABOVEである場合、デコーダはランタイプを解析する必要はない。

1つのセグメント内のピクセルのパレットラン符号化の後、INDEXモードのインデックス値(`palette__idx__idc`)と量子化されたエスケープカラー(`palette__escape__val`)がバイパス符号化される。

【0127】

[00158]＜残差および係数のコーディングの改善＞

【0128】

10

20

30

40

50

[00159] VVCでは、変換係数を符号化するとき、統一された（同じ）ライスパラメータ（RicePara）導出が、abs_remainderおよびdec_abs_levelのシンタックスを伝送するために使用される。唯一の違いは、ベースレベル（baseLevel）が、abs_remainderとdec_abs_levelの符号化に対してそれぞれ4と0に設定されていることである。ライスパラメータは、次のように、ローカルテンプレート内の隣接する5つの変換係数の絶対レベルの合計だけでなく、対応するベースレベルにも基づいて決定される。

[0129]

[00160] RicePara = RiceParTable [max (min (31, sumAbs - 5 * baseLevel), 0)]

10

[0130]

[00161] 言い換えれば、シンタックス要素abs_remainderおよびdec_abs_levelの2値符号語は、隣接する係数のレベル情報に従って適応的に決定される。この符号語の決定はサンプルごとに実行されるため、係数コーディングに対するこの符号語の適応を処理するには追加のロジックが必要である。

[0131]

[00162] 同様に、変換スキップモードの下で残差ブロックを符号化するとき、シンタックス要素abs_remainderの2値符号語は、隣接する残差サンプルのレベル情報に従って適応的に決定される。

[0132]

20

[00163] さらに、残差コーディングまたは変換係数コーディングに関連するシンタックス要素を符号化する場合、確率モデルの選択は、追加のロジックおよび追加のコンテキストモデルを必要とする隣接レベルのレベル情報に依存する。

[0133]

[00164] 現在の設計では、エスケープサンプルの2値化は、3次のExp-Golomb 2値化プロセスを呼び出すことによって導出される。その性能にはさらに改善の余地がある。

[0134]

[00165] 現在のVVCでは、2つの異なるレベルマッピング方式が利用可能であり、それぞれ通常の変換および変換スキップに適用される。各レベルマッピング方式は、異なる条件、マッピング関数、およびマッピング位置に関連付けられている。通常の変換が適用されるブロックの場合、コンテキスト符号化ビン（CCB）の数が制限を超えた後、レベルマッピング方式が使用される。ZeroPosf[n]で示されるマッピング位置、およびAbsLevel[xC][yC]で示されるマッピング結果は、表2に指定されているように導出される。変換スキップが適用されるブロックの場合、コンテキスト符号化ビン（CCB）の数が制限を超える前に、別のレベルマッピング方式が使用される。predCoeffとして示されるマッピング位置、およびAbsLevel[xC][yC]として示されるマッピング結果は、テーブル5に指定されているように導出される。このような統一されていない設計は、標準化の観点からは最適ではない可能性がある。

30

[0135]

40

[00166] HEVCにおける10ビットを超えるプロファイルの場合、1に等しいextended_precision_processing_flagは、拡張ダイナミックレンジが係数解析および逆変換処理に使用されることを指定する。現在のVVCでは、10ビットを超える変換係数の残差コーディングまたは変換スキップコーディングが性能の大幅な低下の原因として報告されている。その性能にはさらに改善の余地がある。

[0136]

[00167] <提案方法>

[0137]

[00168] 本開示では、残差および係数のコーディングの改善のセクションで述べた問題に対処するために、いくつかの方法が提案される。以下の方法は、独立してまたは組み合

50

わせて適用できることに留意されたい。

【0138】

[00169]本開示の第1の態様によれば、残差コーディングにおいて特定のシンタックス要素、例えば、`abs__remainder`を符号化するために2値符号語の固定セットを使用することが提案される。2値符号語は、さまざまな方法を使用して形成できる。いくつかの例示的な方法を以下に挙げる。

【0139】

[00170]まず、現在のVVCで使用されるのと同じ`abs__remainder`の符号語を決定する手順が使用されるが、常に固定ライスパラメータ（例えば、1、2、または3）が選択される。

10

【0140】

[00171]第2に、固定長の2値化。

【0141】

[00172]第3に、切り捨てライス2値化。

【0142】

[00173]第4に、切り捨て2値（TB）2値化プロセス。

【0143】

[00174]第5、k次のExp-Golomb 2値化プロセス（EGk）。

【0144】

[00175]第6の、限定されたk次のExp-Golomb 2値化。

20

【0145】

[00176]本開示の第2の態様によれば、変換係数コーディングにおいて、あるシンタックス要素、例えば、`abs__remainder`および`dec__abs__level`を符号化するために固定セットの符号語を使用することが提案される。2値符号語は、さまざまな方法を使用して形成できる。いくつかの例示的な方法を以下に挙げる。

【0146】

[00177]第1に、現在のVVCで使用されるのと同じ、`abs__remainder`および`dec__abs__level`の符号語を決定するための同じ手順が使用されるが、固定ライスパラメータ、例えば、1、2または3が使用される。現在のVVCで使用されているように、`baseLevel`の値は、`abs__remainder`と`dec__abs__level`で依然として異なり得る（例えば、`abs__remainder`と`dec__abs__level`を符号化する場合、`baseLevel`はそれぞれ4と0に設定される）。

30

【0147】

[00178]第2に、現在のVVCで使用されるのと同じ、`abs__remainder`および`dec__abs__level`の符号語を決定するための同じ手順が使用されるが、固定ライスパラメータ、例えば、1、2または3が使用される。`abs__remainder`と`dec__abs__level`の`baseLevels`の値は同じになるように選択される。例えば、両方とも0を使用するか、両方とも4を使用する。

【0148】

[00179]第3に、固定長の2値化。

【0149】

[00180]第4に、切り捨てライス2値化。

【0150】

[00181]第5に、切り捨て2値（TB）2値化プロセス。

【0151】

[00182]第6に、k次のExp-Golomb 2値化プロセス（EGk）。

【0152】

[00183]第7の限定されたk次のExp-Golomb 2値化

【0153】

40

50

[00184]本開示の第3の側面によれば、残差コーディングまたは係数コーディング（例えば、`abs__level__gt_x__flag`）に関連するシンタックス要素の符号化に単一のコンテキストを使用することが提案されており、隣接する復号レベル情報に基づくコンテキスト選択は削除できる。

【0154】

[00185]本開示の第4の態様によれば、残差コーディングにおいて、特定のシンタックス要素、例えば`abs__remainder`を符号化するために2値符号語の可変セットを使用することが提案され、2値符号語のセットの選択は、現在のブロックの特定の符号化情報、例えば`TB/CB`および／またはスライス、`CU`の予測モード（例えば、`IBC`モードまたはイントラもしくはインター）ならびに／あるいはスライスタイプ（例えば、`IS`スライス、`PS`スライス、または`BS`スライス）に関連付けられた量子化パラメータ（`QP`）に従って決定される。2値符号語の変数セットを導出するには、さまざまな方法を使用できる。いくつかの例示的な方法を以下に示す。

10

【0155】

[00186]第1に、現在の`VVC`で使用されるのと、`abs__remainder`の符号語を決定するための同じ手順が使用されるが、ライスパラメータが異なる。

【0156】

[00187]第2、 k 次の`Exp-Golomb`2値化プロセス（`EGk`）

【0157】

[00188]第3の、限定された k 次の`Exp-Golomb`2値化

20

【0158】

30

40

50

【表 6】

テーブル6. QP値に基づくライスパラメータ決定

<pre> if(QP_{CU} < TH1) { ライスパラメータ = K0 } else if(QP_{CU} < TH2) { ライスパラメータ = K1 } else if(QP_{CU} < TH3) { ライスパラメータ = K2 } else if(QP_{CU} < TH4) { ライスパラメータ = K3 } else { ライスパラメータ = K4 } </pre>	10
	20
	30

【0159】

【00189】第4の態様で説明したのと同じ方法は、変換効率の高いコーディングにも適用できる。本開示の第5の態様によれば、変換係数コーディングにおいて、特定のシンタックス要素、例えば `abs_remainder` と `dec_abs_level` を符号化するために2値符号語の変数セットを使用することが提案され、2値符号語のセットの選択は、現在のブロックの特定の符号化情報、例えば `TB`／`CB` および／またはスライス、`CU` の予測モード（例えば、`IBC` モードまたはイントラもしくはインター）ならびに／あるいはスライスタイプ（例えば、`I` スライス、`P` スライス、または `B` スライス）に関連付けられた量子化パラメータ（`QP`）に従って決定される。ここでも、2値符号語の変数セットを導出するためにさまざまな方法が使用される。いくつかの例示的な方法を以下に示す。

【0160】

【00190】第1に、現在の `VVC` で使用されるのと、`abs_remainder` の符号語を決定するための同じ手順が使用されるが、ライスパラメータが異なる。

【0161】

【00191】第2に、 k 次 `Exp-Golomb` 2値化プロセス（`EGk`）。

【0162】

40

50

[00192]第3に、限定されたk次のExp-Golomb 2値化。

【0163】

[00193]上記のこれらの方法では、異なるライスパラメータを使用して、2値符号語の異なるセットを導出することができる。残差サンプルの特定のブロックについて、使用されるライスパラメータは、隣接レベル情報ではなく、QP_{CU}として示されるCUのQPに従って決定される。具体的な一例をテーブル6に示す。ここで、TH1~TH4は(TH1<TH2<TH3<TH4)を満たす所定の閾値であり、K0~K4は所定のライスパラメータである。実際には、同じロジックが異なる方法で実装され得ることは注目に値する。例えば、特定の方程式またはルックアップテーブルを使用して、テーブル6に示すように、現在のCUのQP値から同じライスパラメータを導出することもできる。

10

【0164】

[00194]本開示の第5の態様によれば、変換係数コーディングおよび／または変換スキップ残差コーディングのシンタックス要素の符号語決定に関連するパラメータおよび／または閾値のセットがビットストリームに伝送される。決定された符号語は、エントロピーコード、例えば算術符号化を通じてシンタックス要素を符号化するときに、2値化符号語として使用される。

【0165】

[00195]パラメータおよび／または閾値のセットは、シンタックス要素の符号語決定に関連するすべてのパラメータおよび閾値の完全なセット、またはサブセットであり得ることに留意されたい。パラメータおよび／または閾値のセットは、ビデオビットストリーム内のさまざまなレベルで伝送できる。例えば、シーケンスレベル（例えば、シーケンスパラメータセット）、ピクチャレベル（例えば、ピクチャパラメータセットおよび／またはピクチャヘッダ）、スライスレベル（例えば、スライスヘッダ）、符号化ツリーユニット（CTU）レベルまたは符号化ユニット（CU）レベルで伝送することができる。

20

【0166】

[00196]一例では、変換スキップ残差コーディングにおけるabs_remainderのシンタックスを符号化するための符号語を決定するために使用されるライスパラメータは、スライスヘッダ、ピクチャヘッダ、PPS、および／またはSPSで伝送される。伝送用のライスパラメータは、CUが変換スキップモードとして符号化され、CUが上述のスライスヘッダ、ピクチャヘッダ、PPSおよび／またはSPSなどに関連付けられている場合に、シンタックスabs_remainderを符号化するための符号語を決定するために使用される。

30

【0167】

[00197]本開示の第6の態様によれば、第1および第2の態様に示される符号語決定に関連するパラメータおよび／または閾値のセットは、変換係数コーディングおよび／または変換スキップ残差コーディングのシンタックス要素に使用される。また、現在のブロックに輝度残差／係数が含まれているか色差残差／係数が含まれているかに応じて、異なるセットを使用できる。決定された符号語は、エントロピーコード、例えば算術符号化を通じてシンタックス要素を符号化するときに、2値化符号語として使用される。

【0168】

[00198]一例では、現在のVVCで使用される変換残差コーディングに関連付けられたabs_remainderの符号語は、輝度ブロックと色差ブロックの両方に使用されるが、異なる固定ライスパラメータが輝度ブロックと色差ブロックによってそれぞれ使用される（例えば、輝度ブロックの場合はK1、色差ブロックの場合はK2、K1とK2は整数である）

40

【0169】

[00199]本開示の第7の態様によれば、変換係数コーディングおよび／または変換スキップ残差コーディングのシンタックス要素の符号語決定に関連するパラメータおよび／または閾値のセットがビットストリームに伝送される。また、輝度ブロックと色差ブロックに対して異なるセットを伝送できる。決定された符号語は、エントロピーコード、例えば

50

算術符号化を通じてシンタックス要素を符号化するときに、2 値化符号語として使用される。上記の態様で説明したのと同じ方法は、パレットモードでのエスケープ値符号化、例えば `palette__escape__val` にも適用できる。

【0170】

【00200】本開示の第8の態様によれば、パレットモードであるエスケープ値を符号化するための2 値符号語の異なるセットを導出するために、異なる k 次の `Exp-Golomb` 2 値化を使用することができる。一例では、エスケープサンプルの所与のブロックについて、使用される `Exp-Golomb` パラメータ、すなわち k の値は、`QP_cu` として示されるブロックの `QP` 値に従って決定される。テーブル6に示されているのと同じ例を、ブロックの所与の `QP` 値に基づいてパラメータ k の値を導出する際に使用することができる。この例では、4つの異なる閾値 (`TH1` から `TH4`) がリストされており、これらの閾値と `QP_cu` に基づいて5つの異なる k 値 ($K0$ から $K4$) が導出されるが、閾値の数は説明のみを目的としていることは注目に値する。実際には、異なる数の閾値を使用して、`QP` 値範囲全体を異なる数の `QP` 値セグメントに分割することができ、各 `QP` 値セグメントに対して、異なる k 値を使用して、パレットモードで符号化されるブロックのエスケープ値を符号化するための対応する2 値符号語を導出することができる。実際には、同じロジックが異なる方法で実装され得ることも注目に値する。例えば、特定の方程式またはルックアップテーブルを使用して、同じライスパラメータを導出することができる。

10

【0171】

【00201】本開示の第9の態様によれば、エスケープサンプルのシンタックス要素の符号語決定に関連するパラメータおよび／または閾値のセットがビットストリームに伝送される。決定された符号語は、エントロピーコーダ、例えば算術符号化を通じてエスケープサンプルのシンタックス要素を符号化するときに、2 値化符号語として使用される。

20

【0172】

【00202】パラメータおよび／または閾値のセットは、シンタックス要素の符号語決定に関連するすべてのパラメータおよび閾値の完全なセット、またはサブセットであり得ることに留意されたい。パラメータおよび／または閾値のセットは、ビデオビットストリーム内のさまざまなレベルで伝送できる。例えば、シーケンスレベル（例えば、シーケンスパラメータセット）で、ピクチャレベル（例えば、ピクチャパラメータセットおよび／またはピクチャヘッダ）、スライスレベル（例えば、スライスヘッダ）、符号化ツリーユニット (`CTU`) レベルまたは符号化ユニット (`CU`) レベルで伝送することができる。

30

【0173】

【00203】本態様による一例では、 k 次の `Exp-Golomb` 2 値化を使用して、パレットモードで `palette__escape__val` のシンタックスを符号化するための符号語を決定し、 k の値がビットストリームでデコーダに伝送される。 k の値は、異なるレベルで伝送されてもよく、例えば、スライスヘッダ、ピクチャヘッダ、`PPS`、および／または `SPS` などでも伝送されてもよい。伝送された `Exp-Golomb` パラメータは、`CU` がパレットモードとして符号化され、`CU` が前述のスライスヘッダ、ピクチャヘッダ、`PPS` および／または `SPS` などに関連付けられている場合に、シンタックス `palette__escape__val` を符号化するための符号語を決定するために使用される。

40

【0174】

【00204】＜変換スキップモードと通常変換モードのレベルマッピングの調和＞

【0175】

【00205】本開示の第10の態様によれば、レベルマッピングを適用するための同じ条件が、変換スキップモードと通常変換モードの両方に使用される。一例では、コンテキスト符号化ビン (`CCB`) の数が変換スキップモードと通常変換モードの両方の制限を超えた後にレベルマッピングを適用することが提案される。別の例では、コンテキスト符号化ビン (`CCB`) の数が変換スキップモードと通常変換モードの両方の制限を超える前にレベルマッピングを適用することが提案されている。

【0176】

50

[00206]本開示の第11の態様によれば、レベルマッピングにおけるマッピング位置を導出する同じ方法が、変換スキップモードと通常変換モードの両方に使用される。一例では、変換スキップモードで使用されるレベルマッピングにおけるマッピング位置の導出方法を通常変換モードにも適用することが提案される。別の例では、通常変換モードで使用されるレベルマッピングにおけるマッピング位置の導出方法を変換スキップモードにも適用することが提案されている。

【0177】

[00207]本開示の第12の態様によれば、同じレベルのマッピング方法が変換スキップモードと通常変換モードの両方に適用される。一例では、変換スキップモードで使用されるレベルマッピング関数を通常変換モードにも適用することが提案される。別の例では、通常変換モードで使用されるレベルマッピング関数を変換スキップモードにも適用することが提案されている。

【0178】

[00208]<残差コーディングにおけるライスパラメータ導出の簡略化>

【0179】

[00209]本開示の第13の態様によれば、Golomb-Riceコードを使用して `abs_remainder/dec_abs_level` のシンタックス要素を符号化する際に、ライスパラメータを導出するルックアップテーブルの代わりに、シフト演算や除算演算などの単純なロジックを使用することが提案されている。現在の開示によれば、テーブル4に指定されているルックアップテーブルは削除されてもよい。一例では、ライスパラメータ `cRiceParam` は、`cRiceParam = (locSumAbs >> n)` として導出され、`n` は正の数、例えば3である。実際には、同じ結果を達成するために他の異なるロジック、例えば2の `n` 乗に等しい値による除算演算を使用できることは注目に値する。VVCドラフトに基づく対応する復号プロセスの例を以下に示す。変更は太字と斜体のフォントで示され、削除された内容は取り消し線のフォントで示されている。

【0180】

10

20

30

40

50

【表 7】

テーブル7. ライスパラメータ導出プロセス

このプロセスへの入力は、ベースレベルの`baseLevel`、カラー成分インデックス`cldx`、現在のピクチャの左上のサンプルを基準とした現在の変換ブロックの左上のサンプルを指定する輝度位置(`x0`, `y0`)、現在の係数スキャン位置(`xC`, `yC`)、変換ブロック幅の2進対数`log2TbWidth`、および変換ブロックの高さの2進対数`log2TbHeight`である。

このプロセスの出力は、ライスパラメータ`cRiceParam`である。

成分インデックス`cldx`と左上の輝度位置(`x0`, `y0`)を持つ変換ブロックの配列`AbsLevel[x][y]`を指定すると、変数`locSumAbs`は次の疑似コードで指定されているように導出される。

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

変数`locSumAbs`を指定すると、ライスパラメータ`cRiceParam`はテーブル4に示される次のように導出される。

cRiceParam = (locSumAbs >> 3)

`baseLevel`が0に等しい場合、変数`ZeroPos[n]`は次のように導出される。

ZeroPos[n] = (QState < 2 ? 1 : 2) << cRiceParam

【0181】

[00210]本開示の第14の態様によれば、Golomb-Riceコードを使用して`abs_remainder/dec_abs_level`のシンタックス要素を符号化する際に、ライスパラメータの導出のためにより少ない隣接位置を使用することが提案される。一例では、`abs_remainder/dec_abs_level`のシンタックス要素を符号化する際のライスパラメータの導出に2つの隣接位置のみを使用することが提案される。VVCDラフトに基づく対応する復号プロセスを以下に示す。変更は太字および斜体のフォントで、削除された内容は取り消し線のフォントで示されている。

【0182】

【表 8】

テーブル8. ライスパラメータ導出プロセス

このプロセスへの入力は、ベースレベルの`baseLevel`、カラー成分インデックス`cldx`、現在のピクチャの左上のサンプルを基準とした現在の変換ブロックの左上のサンプルを指定する輝度位置 $(x0, y0)$ 、現在の係数スキャン位置 (xC, yC) 、変換ブロック幅の2進対数 $\log_2 TbWidth$ 、および変換ブロックの高さの2進対数 $\log_2 TbHeight$ である。

このプロセスの出力は、ライスパラメータ`cRiceParam`である。

成分インデックス`cldx`と左上の輝度位置 $(x0, y0)$ を持つ変換ブロックの配列`AbsLevel[x][y]`を指定すると、変数`locSumAbs`は次の疑似コードで指定されているように導出される。

```
locSumAbs = 0
if( xC < (1 << log2TbWidth) - 1 ) {
    locSumAbs += AbsLevel[ xC + 1 ][ yC ]
}
if( yC < (1 << log2TbHeight) - 1 ) {
    locSumAbs += AbsLevel[ xC ][ yC + 1 ]
}
locSumAbs = Clip3( 0, 31, locSumAbs - baseLevel * 52 )
```

変数`locSumAbs`を指定すると、ライスパラメータ`cRiceParam`がテーブル4に示されるように導出される。

`baseLevel`が0に等しい場合、変数`ZeroPos[n]`は次のように導出される。

```
ZeroPos[ n ] = ( QState < 2 ? 1 : 2 ) << cRiceParam
```

【0183】

[00211]別の例では、`abs_remainder / dec_abs_level`のシンタックス要素を符号化する際のライスパラメータの導出のために、1つの隣接位置のみを使用することが提案される。VVCドラフトに基づく対応する復号プロセスを以下に示す。変更は太字および斜体のフォントで、削除された内容は取り消し線のフォントで示されている。

【0184】

10

20

30

40

50

【表 9】

テーブル9. ライスパラメータ導出プロセス

このプロセスへの入力は、ベースレベルの`baseLevel`、カラー成分インデックス`cldx`、現在のピクチャの左上のサンプルを基準とした現在の変換ブロックの左上のサンプルを指定する輝度位置 $(x0, y0)$ 、現在の係数スキャン位置 (xC, yC) 、変換ブロック幅の2進対数 $\log2TbWidth$ 、および変換ブロックの高さの2進対数 $\log2TbHeight$ である。

このプロセスの出力は、ライスパラメータ`cRiceParam`である。

成分インデックス`cldx`と左上の輝度位置 $(x0, y0)$ を持つ変換ブロックの配列`AbsLevel[x][y]`を指定すると、変数`locSumAbs`は次の疑似コードで指定されているように導出される。

```
locSumAbs = 0
if( xC < (1 << log2TbWidth) - 1 ) {
    locSumAbs += AbsLevel[ xC + 1 ][ yC ]
    if( xC < (1 << log2TbWidth) - 2 )
    }
locSumAbs = Clip3( 0, 31, locSumAbs - baseLevel )
```

変数`locSumAbs`を指定すると、ライスパラメータ`cRiceParam`がテーブル4に示されるように導出される。

`baseLevel`が0に等しい場合、変数`ZeroPos[n]`は次のように導出される。

```
ZeroPos[ n ] = ( QState < 2 ? 1 : 2 ) << cRiceParam
```

【0185】

[00212]本開示の第15の態様によれば、Golomb-Riceコードを使用して`abs_remainder / dec_abs_level`のシンタックス要素を符号化する際に、ライスパラメータを導出するために、`baseLevel`の値に基づいて`locSumAbs`の値を調整するために異なるロジックを使用することが提案されている。一例では、追加のスケールおよびオフセット演算が「 $(locSumAbs - baseLevel * 5) * alpha + beta$ 」の形式で適用される。`alpha`の値が1.5、`beta`の値が1の場合、VVCドラフトに基づく対応する復号プロセスは次のように示される。変更は太字の斜体フォントで、削除された内容は取り消し線のフォントで示されている。

【0186】

10

20

30

40

50

【表 10】

テーブル10. ライスパラメータ導出プロセス

このプロセスへの入力は、ベースレベルのbaseLevel、カラー成分インデックスcldx、現在のピクチャの左上のサンプルを基準とした現在の変換ブロックの左上のサンプルを指定する輝度位置(x0、y0)、現在の係数スキャン位置(xC、yC)、変換ブロック幅の2進対数log2TbWidth、および変換ブロックの高さの2進対数log2TbHeightである。

このプロセスの出力は、ライスパラメータcRiceParamである。

成分インデックスcldxと左上の輝度位置(x0、y0)を持つ変換ブロックの配列AbsLevel[x][y]を指定すると、変数locSumAbsは次の疑似コードで指定されているように導出される。

```
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 ]
}
locSumAbs = Clip3( 0, 31, (locSumAbs - baseLevel * 5) * 1.5 + 1 )
```

変数locSumAbsを指定すると、ライスパラメータcRiceParamがテーブル4に示されるように導出される。

baseLevelが0に等しい場合、変数ZeroPos[n]は次のように導出される。

$ZeroPos[n] = (QState < 2 ? 1 : 2) << cRiceParam$

【0187】

【00213】本開示の第16の態様によれば、Golomb-Riceコードを使用して、abs_remainder/dec_abs_levelのシンタックス要素におけるライスパラメータの導出のためのクリップ演算を除去することが提案される。現在の開示によれば、VVCドラフトの復号プロセスの例は以下のように示されており、変更は太字および斜体のフォントで示され、削除された内容は取り消し線のフォントで示されている。

【0188】

10

20

30

40

50

【表 1 1】

テーブル11. ライスパラメータ導出プロセス

このプロセスへの入力は、ベースレベルの`baseLevel`、カラー成分インデックス`cldx`、現在のピクチャの左上のサンプルを基準とした現在の変換ブロックの左上のサンプルを指定する輝度位置(`x0`, `y0`)、現在の係数スキャン位置(`xC`, `yC`)、変換ブロック幅の2進対数`log2TbWidth`、および変換ブロックの高さの2進対数`log2TbHeight`である。

このプロセスの出力は、ライスパラメータ`cRiceParam`である。

成分インデックス`cldx`と左上の輝度位置(`x0`, `y0`)を持つ変換ブロックの配列`AbsLevel[x][y]`を指定すると、変数`locSumAbs`は次の疑似コードで指定されているように導出される。

```
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 ]
}
locSumAbs = locSumAbs - baseLevel * 5
```

変数`locSumAbs`を指定すると、ライスパラメータ`cRiceParam`はテーブル4に示される次のように導出される。

cRiceParam = (locSumAbs >> 3)

`BaseLevel`が0に等しい場合、変数`ZeroPos[n]`は次のように導出される。

ZeroPos[n] = (QState < 2 ? 1 : 2) << cRiceParam

【0 1 8 9】

【00214】本開示によれば、VVCドラフトの復号プロセスの一例が以下に示されており、変更は太字および斜体のフォントで示され、削除された内容は取り消し線のフォントで示されている。

【0 1 9 0】

10

20

30

40

50

【表 1 2】

テーブル12. ライスパラメータ導出プロセス

このプロセスへの入力は、ベースレベルの`baseLevel`、カラー成分インデックス`cldx`、現在のピクチャの左上のサンプルを基準とした現在の変換ブロックの左上のサンプルを指定する輝度位置(`x0`, `y0`)、現在の係数スキャン位置(`xC`, `yC`)、変換ブロック幅の2進対数`log2TbWidth`、および変換ブロックの高さの2進対数`log2TbHeight`である。

このプロセスの出力は、ライスパラメータ`cRiceParam`である。

成分インデックス`cldx`と左上の輝度位置(`x0`, `y0`)を持つ変換ブロックの配列`AbsLevel[x][y]`を指定すると、変数`locSumAbs`は次の疑似コードで指定されているように導出される。

```
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 ]
}
locSumAbs = Min(31, locSumAbs - baseLevel * 5 )
```

変数`locSumAbs`を指定すると、ライスパラメータ`cRiceParam`がテーブル4に示されるように導出される。

`baseLevel`が0に等しい場合、変数`ZeroPos[n]`は次のように導出される。

`ZeroPos[ n ] = ( QState < 2 ? 1 : 2 ) << cRiceParam`

【0 1 9 1】

【00215】本開示の第17の態様によれば、Golomb-Riceコードを使用して`abs_remainder / dec_abs_level`のシンタックス要素を符号化する際のライスパラメータの導出のために、`locSumAbs`の初期値を0から非ゼロの整数に変更することが提案されている。一例では、初期値1が`locSumAbs`に割り当てられ、VVCドラフトに基づく対応する復号プロセスが以下に示される。変更は太字の斜体フォントで、削除された内容は取り消し線フォントで示されている。

【0 1 9 2】

10

20

30

40

50

【表 1 3】

テーブル13. ライスパラメータ導出プロセス

このプロセスへの入力は、ベースレベルのbaseLevel、カラー成分インデックスcldx、現在のピクチャの左上のサンプルを基準とした現在の変換ブロックの左上のサンプルを指定する輝度位置(x0、y0)、現在の係数スキャン位置(xC、yC)、変換ブロック幅の2進対数log2TbWidth、および変換ブロックの高さの2進対数log2TbHeightである。

このプロセスの出力は、ライスパラメータcRiceParamである。

成分インデックスcldxと左上の輝度位置(x0, y0)を持つ変換ブロックの配列AbsLevel[x][y]を指定すると、変数locSumAbsは次の疑似コードで指定されているように導出される。

```
locSumAbs = 1
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 ]
}
locSumAbs = Clip3( 0, 31, locSumAbs - baseLevel * 5 )
```

変数locSumAbsを指定すると、ライスパラメータcRiceParamがテーブル4に示されるように導出される。

baseLevelが0に等しい場合、変数ZeroPos[n]は次のように導出される。

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

【0 1 9 3】

[00216]本開示の第18の態様によれば、Golomb-Riceコードを使用してabs_remainder/dec_abs_levelのシンタックス要素を符号化する際のライスパラメータの導出には、隣接位置レベル値の合計値の代わりにその最大値を使用することが提案されている。VVCDラフトに基づく対応する復号プロセスの例を以下に示す。変更は太字および斜体のフォントで示され、削除された内容は取り消し線のフォントで示されている。

【0 1 9 4】

10

20

30

40

50

【表 1 4】

テーブル14. ライスパラメータ導出プロセス

このプロセスへの入力は、ベースレベルの`baseLevel`、カラー成分インデックス`cldx`、現在のピクチャの左上のサンプルを基準とした現在の変換ブロックの左上のサンプルを指定する輝度位置(`x0`, `y0`)、現在の係数スキャン位置(`xC`, `yC`)、変換ブロック幅の2進対数`log2TbWidth`、および変換ブロックの高さの2進対数`log2TbHeight`である。

このプロセスの出力は、ライスパラメータ`cRiceParam`である。

成分インデックス`cldx`と左上の輝度位置(`x0`, `y0`)を持つ変換ブロックの配列`AbsLevel[x][y]`を指定すると、変数`locSumAbs`は次の疑似コードで指定されているように導出される。

```
locSumAbs = 0
if( xC < (1 << log2TbWidth) - 1 ) {
    locSumAbs += Max( AbsLevel[ xC + 1 ][ yC ], locSumAbs )
    if( xC < (1 << log2TbWidth) - 2 )
        locSumAbs += Max( AbsLevel[ xC + 2 ][ yC ], locSumAbs )
    if( yC < (1 << log2TbHeight) - 1 )
        locSumAbs += Max( AbsLevel[ xC + 1 ][ yC + 1 ], locSumAbs )      (1494)
}
if( yC < (1 << log2TbHeight) - 1 ) {
    locSumAbs += Max( AbsLevel[ xC ][ yC + 1 ], locSumAbs )
    if( yC < (1 << log2TbHeight) - 2 )
        locSumAbs += Max( AbsLevel[ xC ][ yC + 2 ], locSumAbs )
}
locSumAbs = Clip3( 0, 31, locSumAbs - baseLevel * 5 )
```

変数`locSumAbs`を指定すると、テーブル4に示されるようにライスパラメータ`cRiceParam`が導出される。

`baseLevel`が0に等しい場合、変数`ZeroPos[n]`は次のように導出される。

`ZeroPos[ n ] = ( QState < 2 ? 1 : 2 ) << cRiceParam`

【 0 1 9 5 】

[00217]本開示の第19の態様によれば、Golomb-Riceコードを使用して`abs_remainder / dec_abs_level`のシンタックス要素を符号化する際に、隣接する位置の各`AbsLevel`値とベースレベル値の相対振幅に基づいてライスパラメータを導出することが提案されている。一例では、ライスパラメータは、隣接する位置の`AbsLevel`値のうちのどれだけがベースレベルよりも大きいかに基づいて導出される。VVCドラフトに基づく対応する復号プロセスの例を以下に示す。変更は太字および斜体のフォントで示され、削除された内容は取り消し線のフォントで示されている。

【 0 1 9 6 】

【表 1 5】

テーブル15. ライスパラメータ導出プロセス

このプロセスへの入力は、ベースレベルの`baseLevel`、カラー成分インデックス`cldx`、現在のピクチャの左上のサンプルを基準とした現在の変換ブロックの左上のサンプルを指定する輝度位置(`x0`、`y0`)、現在の係数スキャン位置(`xC`、`yC`)、変換ブロック幅の2進対数`log2TbWidth`、および変換ブロックの高さの2進対数`log2TbHeight`である。

このプロセスの出力は、ライスパラメータ`cRiceParam`である。

成分インデックス`cldx`と左上の輝度位置(`x0`、`y0`)を持つ変換ブロックの配列`AbsLevel[x][y]`を指定すると、変数`locSumAbs`は次の疑似コードで指定されているように導出される。

```
locSumAbs = 0
if( xC < (1 << log2TbWidth) - 1 ) {
    locSumAbs += (AbsLevel[ xC + 1 ][ yC ] > baseLevel? 1:0)
    if( xC < (1 << log2TbWidth) - 2 )
        locSumAbs += (AbsLevel[ xC + 2 ][ yC ] > baseLevel? 1:0)
    if( yC < (1 << log2TbHeight) - 1 )
        locSumAbs += (AbsLevel[ xC + 1 ][ yC + 1 ] > baseLevel? 1:0)      (1494)
}
if( yC < (1 << log2TbHeight) - 1 ) {
    locSumAbs += (AbsLevel[ xC ][ yC + 1 ] > baseLevel? 1:0)
    if( yC < (1 << log2TbHeight) - 2 )
        locSumAbs += (AbsLevel[ xC ][ yC + 2 ] > baseLevel? 1:0)
}
```

変数`locSumAbs`を指定すると、ライスパラメータ`cRiceParam`は**テーブル4**に示される次のように導出される。

cRiceParam = locSumAbs

`baseLevel`が0に等しい場合、変数`ZeroPos[n]`は次のように導出される。

`ZeroPos[ n ] = ( QState < 2 ? 1 : 2 ) << cRiceParam`

【 0 1 9 7 】

【00218】別の例では、ライスパラメータは、`AbsLevel`値がベースレベルより大きい隣接位置の(`AbsLevel - baseLevel`)値の合計に基づいて導出される。`VVC`ドラフトに基づく対応する復号プロセスの例を以下に示す。変更は太字および斜体のフォントで示され、削除された内容は取り消し線のフォントで示されている。

【 0 1 9 8 】

10

20

30

40

50

【表 1 6】

テーブル16. ライスパラメータ導出プロセス

このプロセスへの入力は、ベースレベルの`baseLevel`、カラー成分インデックス`cldx`、現在のピクチャの左上のサンプルを基準とした現在の変換ブロックの左上のサンプルを指定する輝度位置(`x0`、`y0`)、現在の係数スキャン位置(`xC`、`yC`)、変換ブロック幅の2進対数`log2TbWidth`、および変換ブロックの高さの2進対数`log2TbHeight`である。

このプロセスの出力は、ライスパラメータ`cRiceParam`である。

成分インデックス`cldx`と左上の輝度位置(`x0`、`y0`)を持つ変換ブロックの配列`AbsLevel[x][y]`を指定すると、変数`locSumAbs`は次の疑似コードで指定されているように導出される。

```
locSumAbs = 0
if( xC < (1 << log2TbWidth) - 1 ) {
    locSumAbs += Max( 0 , AbsLevel[ xC + 1 ][ yC ] - baseLevel)
    if( xC < (1 << log2TbWidth) - 2 )
        locSumAbs += Max( 0 , AbsLevel[ xC + 2 ][ yC ] - baseLevel)
    if( yC < (1 << log2TbHeight) - 1 )
        locSumAbs += Max( 0 , AbsLevel[ xC + 1 ][ yC + 1 ] - baseLevel)      (1494)
}
if( yC < (1 << log2TbHeight) - 1 ) {
    locSumAbs += Max( 0 , AbsLevel[ xC ][ yC + 1 ] - baseLevel)
    if( yC < (1 << log2TbHeight) - 2 )
        locSumAbs += Max( 0 , AbsLevel[ xC ][ yC + 2 ] - baseLevel)
}
locSumAbs = Min( 31, locSumAbs )
```

変数`locSumAbs`を指定すると、ライスパラメータ`cRiceParam`がテーブル4に示されるように導出される。

`baseLevel`が0に等しい場合、変数`ZeroPos[n]`は次のように導出される。

`ZeroPos[ n ] = ( QState < 2 ? 1 : 2 ) << cRiceParam`

【 0 1 9 9 】

【00219】本開示によれば、VVCドラフトの復号プロセスの一例が以下に示されており、変更は太字および斜体のフォントで示され、削除された内容を取り消し線のフォントで示されている。

【 0 2 0 0 】

10

20

30

40

50

【表 17】

テーブル17. ライスパラメータ導出プロセス

このプロセスへの入力は、ベースレベルの $baseLevel$ 、カラー成分インデックス $cldx$ 、現在のピクチャの左上のサンプルを基準とした現在の変換ブロックの左上のサンプルを指定する輝度位置 $(x0, y0)$ 、現在の係数スキャン位置 (xC, yC) 、変換ブロック幅の2進対数 $\log2TbWidth$ 、および変換ブロックの高さの2進対数 $\log2TbHeight$ である。

このプロセスの出力は、ライスパラメータ $cRiceParam$ である。

成分インデックス $cldx$ と左上の輝度位置 $(x0, y0)$ を持つ変換ブロックの配列 $AbsLevel[x][y]$ を指定すると、変数 $locSumAbs$ は次の疑似コードで指定されているように導出される。

```
locSumAbs = 0
if( xC < (1 << log2TbWidth) - 1 ) {
    locSumAbs += AbsLevel[ xC + 1 ][ yC ] - baseLevel
    if( xC < (1 << log2TbWidth) - 2 )
        locSumAbs += AbsLevel[ xC + 2 ][ yC ] - baseLevel
    if( yC < (1 << log2TbHeight) - 1 )
        locSumAbs += AbsLevel[ xC + 1 ][ yC + 1 ] - baseLevel
}
if( yC < (1 << log2TbHeight) - 1 ) {
    locSumAbs += AbsLevel[ xC ][ yC + 1 ] - baseLevel
    if( yC < (1 << log2TbHeight) - 2 )
        locSumAbs += AbsLevel[ xC ][ yC + 2 ] - baseLevel
}
locSumAbs = Clip3( 0, 31, locSumAbs)
(1494)
```

変数 $locSumAbs$ を指定すると、ライスパラメータ $cRiceParam$ がテーブル4に示されるように導出される。

$BaseLevel$ が0に等しい場合、変数 $ZeroPos[n]$ は次のように導出される。

$ZeroPos[n] = (QState < 2 ? 1 : 2) << cRiceParam$

【0201】

【00220】＜残差コーディングにおけるレベルマッピング位置導出の簡略化＞

【0202】

【00221】本開示の第20の態様によれば、 $ZeroPos[n]$ の導出から $QState$ を除去し、 $ZeroPos[n]$ が $cRiceParam$ のみから導出されるようにすることが提案される。VVCドラフトに基づく対応する復号プロセスの例を以下に示す。変更は太字および斜体のフォントで示され、削除された内容は取り消し線のフォントで示されている。

【0203】

【表 18】

テーブル18. ライスパラメータ導出プロセス

このプロセスへの入力は、ベースレベルの`baseLevel`、カラー成分インデックス`cldx`、現在のピクチャの左上のサンプルを基準とした現在の変換ブロックの左上のサンプルを指定する輝度位置 $(x0, y0)$ 、現在の係数スキャン位置 (xC, yC) 、変換ブロック幅の2進対数 $\log_2 TbWidth$ 、および変換ブロックの高さの2進対数 $\log_2 TbHeight$ である。

このプロセスの出力は、ライスパラメータ`cRiceParam`である。

成分インデックス`cldx`と左上の輝度位置 $(x0, y0)$ を持つ変換ブロックの配列`AbsLevel[x][y]`を指定すると、変数`locSumAbs`は次の疑似コードで指定されているように導出される。

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

変数`locSumAbs`を指定すると、ライスパラメータ`cRiceParam`がテーブル4に示されるように導出される。

`baseLevel`が0に等しい場合、変数`ZeroPos[n]`は次のように導出される。

`ZeroPos[ n ] = 2 << cRiceParam`

【0204】

【00222】本開示の第21の態様によれば、`locSumAbs`の値に基づいて`ZeroPos[n]`を導出することが提案される。VVCドラフトに基づく対応する復号プロセスの例を以下に示す。変更は太字および斜体のフォントで示され、削除された内容は取り消し線のフォントで示されている。

【0205】

10

20

30

40

50

【表 19】

テーブル19. ライスパラメータ導出プロセス

このプロセスへの入力は、ベースレベルの`baseLevel`、カラー成分インデックス`cldx`、現在のピクチャの左上のサンプルを基準とした現在の変換ブロックの左上のサンプルを指定する輝度位置(`x0`, `y0`)、現在の係数スキャン位置(`xC`, `yC`)、変換ブロック幅の2進対数`log2TbWidth`、および変換ブロックの高さの2進対数`log2TbHeight`である。

このプロセスの出力は、ライスパラメータ`cRiceParam`である。

成分インデックス`cldx`と左上の輝度位置(`x0`, `y0`)を持つ変換ブロックの配列`AbsLevel[x][y]`を指定すると、変数`locSumAbs`は次の疑似コードで指定されているように導出される。

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

変数`locSumAbs`を指定すると、ライスパラメータ`cRiceParam`がテーブル4に示されるように導出される。

`baseLevel`が0に等しい場合、変数`ZeroPos[n]`は次のように導出される。

`ZeroPos[ n ] = locSumAbs`

【0206】

[00223]本開示の第22の態様によれば、隣接する位置の`AbsLevel`の値に基づいて`ZeroPos[n]`を導出することが提案される。一例では、`ZeroPos[n]`は、`AbsLevel[xC+1][yC]`および`AbsLevel[xC][yC+1]`のうちの最大値に基づいて導出される。VVCDラフトに基づく対応する復号プロセスの例を以下に示す。変更は太字および斜体のフォントで示され、削除された内容は取り消し線のフォントで示されている。

【0207】

10

20

30

40

50

【表 2 0】

テーブル20. ライスパラメータ導出プロセス

このプロセスへの入力は、ベースレベルのbaseLevel、カラー成分インデックスcldx、現在のピクチャの左上のサンプルを基準とした現在の変換ブロックの左上のサンプルを指定する輝度位置(x0、y0)、現在の係数スキャン位置(xC、yC)、変換ブロック幅の2進対数log2TbWidth、および変換ブロックの高さの2進対数log2TbHeightである。

このプロセスの出力は、ライスパラメータcRiceParamである。

成分インデックスcldxと左上の輝度位置(x0, y0)を持つ変換ブロックの配列AbsLevel[x][y]を指定すると、変数locSumAbsは次の疑似コードで指定されているように導出される。

```
locSumAbs = 0
ZeroPos[ n ] = 1
if( xC < (1 << log2TbWidth) - 1 ) {
    locSumAbs += AbsLevel[ xC + 1 ][ yC ]
    ZeroPos[ n ] = Max( AbsLevel[ xC + 1 ][ yC ], ZeroPos[ n ] )
    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 ]
    ZeroPos[ n ] = Max( AbsLevel[ xC ][ yC + 1 ], ZeroPos[ n ] )
    if( yC < (1 << log2TbHeight) - 2 )
        locSumAbs += AbsLevel[ xC ][ yC + 2 ]
}
locSumAbs = Clip3( 0, 31, locSumAbs - baseLevel * 5 )
```

変数locSumAbsを指定すると、ライスパラメータcRiceParamがテーブル4に示されるように導出される。

baseLevelが0に等しい場合、変数ZeroPos[n]は次のように導出される。

ZeroPos[n] = ZeroPos[n] * 1.25 + 1

【 0 2 0 8】

【00224】本開示の第23の態様によれば、隣接する位置のすべてのAbsLevel値の最大値に基づいて、cRiceParamおよびZeroPos[n]の両方を導出することが提案される。VVCDラフトに基づく対応する復号プロセスの例を以下に示す。変更は太字および斜体のフォントで示され、削除された内容は取り消し線のフォントで示されている。

【 0 2 0 9】

10

20

30

40

【表 2 1】

テーブル21. ライスパラメータ導出プロセス

このプロセスへの入力は、ベースレベルのbaseLevel、カラー成分インデックスcldx、現在のピクチャの左上のサンプルを基準とした現在の変換ブロックの左上のサンプルを指定する輝度位置(x0, y0)、現在の係数スキャン位置(xC, yC)、変換ブロック幅の2進対数log2TbWidth、および変換ブロックの高さの2進対数log2TbHeightである。

このプロセスの出力は、ライスパラメータcRiceParamである。

成分インデックスcldxと左上の輝度位置(x0, y0)を持つ変換ブロックの配列AbsLevel[x][y]を指定すると、変数locSumAbsは次の疑似コードで指定されているように導出される。

```
locSumAbs = 0
if( xC < (1 << log2TbWidth) - 1 ) {
    locSumAbs = Max( AbsLevel[ xC + 1 ][ yC ], locSumAbs )
    if( xC < (1 << log2TbWidth) - 2 )
        locSumAbs = Max( AbsLevel[ xC + 2 ][ yC ], locSumAbs )
    if( yC < (1 << log2TbHeight) - 1 )
        locSumAbs = Max( AbsLevel[ xC + 1 ][ yC + 1 ], locSumAbs )
}
if( yC < (1 << log2TbHeight) - 1 ) {
    locSumAbs = Max( AbsLevel[ xC ][ yC + 1 ], locSumAbs )
    if( yC < (1 << log2TbHeight) - 2 )
        locSumAbs = Max( AbsLevel[ xC ][ yC + 2 ], locSumAbs )
}
locSumAbs = Max ( 0, locSumAbs - baseLevel )
```

変数locSumAbsを指定すると、ライスパラメータcRiceParamは次のように導出される。

cRiceParam = **Min**((locSumAbs >> 2) , 3)

baseLevelが0に等しい場合、変数ZeroPos[n]は次のように導出される。

ZeroPos[n] = **locSumAbs**

【 0 2 1 0】

[00225]上記の態様で説明したのと同じ方法は、変換スキップモードの残差コーディングにおけるpredCoeffの導出にも適用できる。一例では、変数predCoeffは次のように導出される。

【 0 2 1 1】

[00226] **predCoeff** = **Max** (**absLeftCoeff** , **absAboveCoeff**) + 1

【 0 2 1 2】

[00227] <変換係数の残差コーディング>

【0213】

[00228]本開示では、「残差および係数のコーディングの改善」セクションで指摘した問題に対処するために、残差コーディングの既存の設計を簡素化および／またはさらに改善する方法が提供される。一般に、本開示で提案される技術の主な特徴は、次のように要約される。

【0214】

[00229]第1に、現在の設計に基づいて、正規の残差コーディングの下で使用されるライスパラメータ導出を調整する。

【0215】

[00230]第2に、正規の残差コーディングの下で使用される2値方法を変更する。

【0216】

[00231]第3に、正規の残差コーディングの下で使用されるライスパラメータ導出を変更する。

【0217】

[00232]＜現在の設計に基づく残差コーディングにおけるライスパラメータ導出＞

【0218】

[00233]本開示の第24の態様によれば、残差コーディングにおいて、特定のシンタックス要素、例えば `abs_remainder` / `dec_abs_level` を符号化するためにライスパラメータ導出の可変方法を使用することが提案され、選択は現在のブロックの特定の符号化情報、例えば `TB` / `CB` および／またはスライス／プロファイルに関連付けられた量子化パラメータもしくは符号化ビット深度に従って、ならびに／あるいは `TB` / `CB` / スライス / ピクチャ / シーケンスレベルに関連付けられた新しいフラグ、例えば `extended_precision_processing_flag` に従って決定される。ライスパラメータを導出するためにさまざまな方法を使用できる。いくつかの例示的な方法を以下に列挙する。

【0219】

[00234]第1に、 $cRiceParam = (cRiceParam \ll a) + (cRiceParam \gg b) + c$ 、ここで a 、 b および c は正の数、例えば $\{a, b, c\} = \{1, 1, 0\}$ である。実際には、同じ結果を達成するために他の異なるロジック、例えば2の n 乗に等しい値による乗算演算を使用できることは注目に値する。

【0220】

[00235]第2に、 $cRiceParam = (cRiceParam \ll a) + b$ 、ここで a および b は正の数、例えば $\{a, b\} = \{1, 1\}$ である。実際には、同じ結果を達成するために他の異なるロジック、例えば2の n 乗に等しい値による乗算演算を使用できることは注目に値する。

【0221】

[00236]第3に、 $cRiceParam = (cRiceParam * a) + b$ 、ここで a および b は正の数、例えば $\{a, b\} = \{1.5, 0\}$ である。実際には、同じ結果を達成するために他の異なるロジック、例えば2の n 乗に等しい値による乗算演算を使用できることは注目に値する。

【0222】

[00237] *VVC* ドラフトに基づく対応する復号プロセスの例を以下に示す。変更は太字および斜体のフォントで示され、削除された内容は取り消し線のフォントで示される。*VVC* ドラフトへの変更は、テーブル22に太字および斜体のフォントで示されている。実際には、同じロジックが異なる方法で実装され得ることは注目に値する。例えば、特定の方程式またはルックアップテーブルを使用して、現在の *CU* / シーケンスの *BitDepth* 値から同じライスパラメータを導出することもできる。

【0223】

10

20

30

40

50

【表 2 2 - 1】

テーブル22. ライスパラメータ導出プロセス

このプロセスへの入力は、ベースレベルの`baseLevel`、カラー成分インデックス`cldx`、現在のピクチャの左上のサンプルを基準とした現在の変換ブロックの左上のサンプルを指定する輝度位置 $(x0, y0)$ 、現在の係数スキャン位置 (xC, yC) 、変換ブロック幅の2進対数 $\log_2 TbWidth$ 、および変換ブロックの高さの2進対数 $\log_2 TbHeight$ である。

このプロセスの出力は、ライスパラメータ`cRiceParam`である。

成分インデックス`cldx`と左上の輝度位置 $(x0, y0)$ を持つ変換ブロックの配列`AbsLevel[x][y]`を指定すると、変数`locSumAbs`は次の疑似コードで指定されているように導出される。

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

変数`locSumAbs`を指定すると、ライスパラメータ`cRiceParam`がテーブル4に示されるように導出される。

*extended_precision_processing_flag*が1に等しい場合、ライスパラメータ`cRiceParam`は次のように指定する。

```
if(BitDepth < 11)
{
    cRiceParam = cRiceParam
}
```

10

20

30

40

50

【表 2 2 - 2】

```

else if(BitDepth <13)
{
    cRiceParam = cRiceParam + (cRiceParam >> 1)
}
else if(BitDepth <15)
{
    cRiceParam = cRiceParam << 1
}
else
{
    cRiceParam = cRiceParam << 1+ (cRiceParam >> 1)
}

```

baseLevelが0に等しい場合、変数ZeroPos[n]は次のように導出される。

$\text{ZeroPos}[n] = (\text{QState} < 2 ? 1 : 2) \ll \text{cRiceParam}$

【0 2 2 4】

[00238]別の例では、*BitDepth*が所定の閾値（例えば、10、11、12、13、14、15、または16）以上である場合、ライスパラメータ*cRiceParam*は次のように導出される： $\text{cRiceParam} = (\text{cRiceParam} \ll a) + (\text{cRiceParam} \gg b) + c$ 、ここで、*a*、*b*、*c*は正の数、例えば1である。VVCドラフトに基づく対応する復号プロセスを以下に示す。変更は太字および斜体のフォントで、削除された内容は取り消し線のフォントで示されている。VVCドラフトへの変更は、テーブル23に太字および斜体のフォントで示されている。実際には、同じロジックが異なる方法で実装され得ることは注目に値する。例えば、特定の方程式またはルックアップテーブルを使用して、現在のCU／シーケンスの*BitDepth*値から同じライスパラメータを導出することもできる。

【0 2 2 5】

10

20

30

40

50

【表 2 3 - 1】

テーブル23. ライスパラメータ導出プロセス

このプロセスへの入力は、ベースレベルの`baseLevel`、カラー成分インデックス`cldx`、現在のピクチャの左上のサンプルを基準とした現在の変換ブロックの左上のサンプルを指定する輝度位置 $(x0, y0)$ 、現在の係数スキャン位置 (xC, yC) 、変換ブロック幅の2進対数 $\log2TbWidth$ 、および変換ブロックの高さの2進対数 $\log2TbHeight$ である。

このプロセスの出力は、ライスパラメータ`cRiceParam`である。

成分インデックス`cldx`と左上の輝度位置 $(x0, y0)$ を持つ変換ブロックの配列`AbsLevel[x][y]`を指定すると、変数`locSumAbs`は次の疑似コードで指定されているように導出される。

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

変数`locSumAbs`を指定すると、テーブル4に示されるようにライスパラメータ`cRiceParam`が導出される。

ライスパラメータ`cRiceParam`は以下のように指定する。

```
if( BitDepth < 11 )
{
    cRiceParam = cRiceParam
}
else if( BitDepth < 13 )
{
```

10

20

30

40

50

【表 2 3 - 2】

<pre> cRiceParam = cRiceParam + (cRiceParam >> 1) } else if(BitDepth < 15) { cRiceParam = cRiceParam << 1 } else { cRiceParam = cRiceParam << 1 + (cRiceParam >> 1) } </pre>	10
BaseLevelが0に等しい場合、変数ZeroPos[n]は次のように導出される。	
<pre> ZeroPos[n] = (QState < 2 ? 1 : 2) << cRiceParam </pre>	20

【0 2 2 6】

[00239] < 10ビットを超えるプロファイルの残差コーディングにおける2値化法>

【0 2 2 7】

[00240] 本開示の第25の態様によれば、残差コーディングにおいて、特定のシンタックス要素、例えば `abs_remainder / dec_abs_level` を符号化するために2値符号語の変数セットを使用することが提案され、選択は現在のブロックの特定の符号化情報、例えば `TB / CB` および／またはスライス／プロファイルに関連付けられた量子化パラメータもしくは符号化ビット深度に従って、ならびに／あるいは `TB / CB / スライス / ピクチャ / シーケンスレベル` に関連付けられた新しいフラグ、例えば `extended_precision_processing_flag` に従って決定される。2値符号語の変数セットを導出するには、さまざまな方法を使用できる。いくつかの例示的な方法を以下に示す。

【0 2 2 8】

[00241] まず、現在のVVCで使用されるのと、`abs_remainder`の符号語を決定するための同じ手順が使用されるが、常に固定ライスパラメータ（例えば、2、3、4、5、6、7または8）が選択される。固定値は、現在のブロックの特定の符号化情報、例えば、`TB / CB` および／またはスライス／プロファイルに関連付けられた量子化パラメータもしくは符号化ビット深度に従って、ならびに／あるいは `TB / CB / スライス / ピクチャ / シーケンスレベル` に関連付けられたシンタックス要素、例えば、`rice_parameter_value` に従って、異なる条件では異なり得る。具体的な一例をテーブル24に示す。ここで、`TH1 ~ TH4` は (`TH1 < TH2 < TH3 < TH4`) を満たす所定の閾値であり、`K0 ~ K4` は所定のライスパラメータである。実際には、同じロジックが異なる方法で実装され得ることは注目に値する。例えば、特定の方程式またはルックアップテーブルを使用して、テーブル24に示すように、現在のCU／シーケンスの `BitDepth` 値から同じライスパラメータを導出することもできる。

【0 2 2 9】

[00242] 第2に、固定長の2値化。

【0 2 3 0】

10

20

30

40

50

[00243]第3に、切り捨てライス2値化。

【0231】

[00244]第4に、切り捨て2値(TB)2値化プロセス。

【0232】

[00245]第5に、k次のExp-Golomb2値化プロセス(EGk)。

【0233】

[00246]第6に、限定されたk次のExp-Golomb2値化

【0234】

【表24】

テーブル24. ビット深度に基づいたライスパラメータの決定

```

if(BitDepth < TH1)
{
    ライスパラメータ = K0
}
else if(BitDepth < TH2)
{
    ライスパラメータ = K1
}
else if(BitDepth < TH3)
{
    ライスパラメータ = K2
}
else if(BitDepth < TH4)
{
    ライスパラメータ = K3
}
else
{
    ライスパラメータ = K4
}

```

【0235】

[00247]一例では、新しいフラグ、例えば `extended_precision_processing_flag` が1に等しい場合、ライスパラメータ `cRiceParam` は `n` に固定される。ここで、`n` は正の数（例えば、2、3、4、5、6、7または8）である。条件により固定値が異なり得る。VVCドラフトに基づく対応する復号プロセスの例を以下に示す。変更は太字および斜体のフォントで示され、削除された内容は取り消し線のフォントで示されている。VVCドラフトへの変更は、テーブル25に太字および斜体のフォントで示されている。

【0236】

10

20

30

40

50

【表 2 5】

テーブル25. ライスパラメータ導出プロセス

ライスパラメータcRiceParamは次のように導出される。

- transform__skip__flag[x0][y0][cldx]が1に等しい場合、かつ sh__ts__residual__coding__disabled__flagは0に等しい場合、ライスパラメータcRiceParamは1に等しく設定される。
- extended__precision__processing__flagが1に等しい場合、ライスパラメータcRiceParamは6に設定される。
- それ以外の場合、ライスパラメータcRiceParamは、変数baseLevelを4に設定して、カラー成分インデックスcldx、輝度位置(x0, y0)、現在の係数スキャン位置(xC, yC)、変換ブロック幅の2進対数log2Tbwidth、および変換ブロック高さの2進対数log2TbHeightを入力としてテーブル3に指定されているabs__remainder[]のライスパラメータ導出プロセスを呼び出すことによって導出される。

10

【0 2 3 7】

【00248】別の例では、新しいフラグ、例えばextended__precision__processing__flagが1に等しい場合、abs__remainder/de c__abs__levelのシンタックス要素を符号化する際にライスパラメータに対して1つの固定値のみを使用することが提案される。VVCドラフトに基づく対応する復号プロセスを以下に示す。変更は太字および斜体のフォントで、削除された内容は取り消し線のフォントで示されている。VVCドラフトへの変更は、テーブル26に太字および斜体のフォントで示されている。

20

【0 2 3 8】

【表 2 6】

テーブル26. ライスパラメータ導出プロセス

ライスパラメータcRiceParamは次のように導出される。

- extended__precision__processing__flagが1に等しい場合、ライスパラメータcRiceParamは7に設定される。
- transform__skip__flag[x0][y0][cldx]が1に等しく、sh__ts__residual__coding__disabled__flagが0に等しい場合、ライスパラメータcRiceParamは1に等しく設定される。
- それ以外の場合、ライスパラメータcRiceParamは、変数baseLevelを4に設定して、カラー成分インデックスcldx、輝度位置(x0, y0)、現在の係数スキャン位置(xC, yC)、変換ブロック幅の2進対数log2Tbwidth、および変換ブロック高さの2進対数log2TbHeightを入力としてテーブル3に指定されているabs__remainder[]のライスパラメータ導出プロセスを呼び出すことによって導出される。

30

40

【0 2 3 9】

【00249】さらに別の例では、BitDepthが所定の閾値（例えば、10、11、12、13、14、15、または16）以上である場合、ライスパラメータcRiceParamはnに固定される。ここで、nは正の数、例えば、4、5、6、7、または8である。条件により固定値が異なり得る。VVCドラフトに基づく対応する復号プロセスの例

50

を以下に示す。ここで、 TH は、所定の閾値（例えば、10、11、12、13、14、15、または16）である。 VVC ドラフトへの変更はテーブル27に太字および斜体で示されており、変更は太字および斜体で示され、削除された内容は取り消し線で示されている。

【0240】

【表27】

テーブル27. ライスパラメータ導出プロセス

ライスパラメータ $cRiceParam$ は次のように導出される。

- $transform_skip_flag[x0][y0][cldx]$ が1に等しい場合、かつ
 $sh_ts_residual_coding_disabled_flag$ は0に等しい場合、ライスパラメータ $cRiceParam$ は1に等しく設定される。
- $BitDepth$ が TH より大きい場合、ライスパラメータ $cRiceParam$ は6に設定される。
- それ以外の場合、ライスパラメータ $cRiceParam$ は、変数 $baseLevel$ を4に設定して、カラー成分インデックス $cldx$ 、輝度位置($x0$, $y0$)、現在の係数スキャン位置(xC , yC)、変換ブロック幅の2進対数 $\log_2 Tbwidht$ 、および変換ブロック高さの2進対数 $\log_2 TbHeight$ を入力としてテーブル3に指定されている $abs_remainder[]$ のライスパラメータ導出プロセスを呼び出すことによって導出される。

【0241】

[00250]さらに別の例では、 $BitDepth$ が所定の閾値（例えば、10、11、12、13、14、15、または16）より大きい場合、 $abs_remainder / dec_abs_level$ のシンタックス要素を符号化する際に、ライスパラメータに対して1つの固定値のみを使用することが提案される。 VVC ドラフトに基づく対応する復号プロセスを以下に示し、 TH は所定の閾値（例えば、10、11、12、13、14、15、または16）であり、変更は太字および斜体のフォントで示され、削除された内容は取り消し線のフォントで示される。 VVC ドラフトへの変更は、テーブル28に太字および斜体のフォントで示されている。

【0242】

10

20

30

40

50

【表 2 8】

テーブル28. ライスパラメータ導出プロセス

ライスパラメータcRiceParamは次のように導出される。

- *BitDepth*が*TH*より大きい場合、ライスパラメータ*cRiceParam*は7に設定される。
- *transform_skip_flag*[*x0*][*y0*][*cldx*]が1に等しい場合、かつ
*sh_ts_residual_coding_disabled_flag*は0に等しい場合、ライスパラメータ*cRiceParam*は1に等しく設定される。
- それ以外の場合、ライスパラメータ*cRiceParam*は、変数*baseLevel*を4に設定して、カラー成分インデックス*cldx*、輝度位置(*x0*, *y0*)、現在の係数スキャン位置(*xC*, *yC*)、変換ブロック幅の2進対数log2*Tbwidth*、および変換ブロック高さの2進対数log2*TbHeight*を入力としてテーブル3に指定されている*abs_remainder*[]のライスパラメータ導出プロセスを呼び出すことによって導出される。

10

【0 2 4 3】

20

【00251】<残差コーディングにおけるライスパラメータ導出>

【0 2 4 4】

【00252】本開示の第26の態様によれば、残差コーディングにおいて、特定のシンタックス要素、例えば*abs_remainder/dec_abs_level*を符号化するためにライスパラメータ導出の可変方法を使用することが提案され、選択は現在のブロックの特定の符号化情報、例えば*TB/CB*および/またはスライス/プロファイルに関連付けられた量子化パラメータもしくは符号化ビット深度に従って、ならびに/あるいは*TB/CB/スライス/ピクチャ/シーケンスレベル*に関連付けられた新しいフラグ、例えば、*extended_precision_processing_flag*に従って決定される。ライスパラメータを導出するためにさまざまな方法を使用できる。いくつかの例示的な方法を以下に示す。

30

【0 2 4 5】

【00253】第1に、カウンタを使用してライスパラメータを導出することが提案される。カウンタは、符号化された係数の値と、現在のブロックの特定の符号化された情報、例えば、成分IDに従って決定される。具体的な例の1つは、*riceParameter=counter/a*である。ここで、*a*は正の数（例えば、4）であり、2つのカウンタ（輝度/色差で分割）を維持する。これらのカウンタは、各スライスの開始時に0にリセットされる。符号化されると、これがサブTU内で符号化された第1の係数である場合、次のようにカウンタが更新される。

```
if (coeffValue >= (3 << rice)) counter++
if (((coeffValue << 1) < (1 << riceParameter)) &
    & (counter > 0)) counter--;
```

40

【0 2 4 6】

【00254】第2に、VVCにおけるライスパラメータの導出においてシフト演算を追加することが提案される。シフトは、符号化係数の値に応じて決定される。VVCドラフトに基づく対応する復号プロセスの例を以下に示す。シフトは方法1のカウンタに従って決定され、変更は太字および斜体のフォントで、削除された内容は取り消し線のフォントで示される。VVCドラフトへの変更は、テーブル29に太字および斜体のフォントで示されている。

【0 2 4 7】

50

【表 2 9】

テーブル29. ライスパラメータ導出プロセス

このプロセスへの入力は、ベースレベルのbaseLevel、カラー成分インデックスcldx、現在のピクチャの左上のサンプルを基準とした現在の変換ブロックの左上のサンプルを指定する輝度位置(x0、y0)、現在の係数スキャン位置(xC、yC)、変換ブロック幅の2進対数log2TbWidth、および変換ブロックの高さの2進対数log2TbHeightである。

このプロセスの出力は、ライスパラメータcRiceParamである。

成分インデックスcldxと左上の輝度位置(x0、y0)を持つ変換ブロックの配列AbsLevel[x][y]を指定すると、変数locSumAbsは次の疑似コードで指定されているように導出される。

Shift = max(0, counter/4 - 2)

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]

(1494)

}

if(yC < (1 << log2TbHeight) - 1) {

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

if(yC < (1 << log2TbHeight) - 2)

locSumAbs += AbsLevel[xC][yC + 2]

}

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

変数locSumAbsを指定すると、テーブル4に示されるようにライスパラメータcRiceParamが導出される。

cRiceParam = ***cRiceParam*** + ***Shift***

baseLevelが0に等しい場合、変数ZeroPos[n]は次のように導出される。

ZeroPos[n] = (QState < 2 ? 1 : 2) << cRiceParam

【 0 2 4 8】

【00255】第1に、VV Cにおけるライスパラメータの導出においてシフト演算を追加することが提案される。シフトは、現在のブロックの特定の符号化情報、例えばTB / CBおよび／またはスライスプロファイル（例えば14ビットプロファイルまたは16ビットプロファイル）に関連する符号化ビット深度に従って決定される。VV Cドラフトに基づく対応する復号プロセスの例を以下に示す。シフトは方法1のカウンタに従って決定され、変更は太字および斜体のフォントで、削除された内容は取り消し線のフォントで示される。VV Cドラフトへの変更は、テーブル30に太字および斜体のフォントで示されている。

【 0 2 4 9 】

【表 3 0】

テーブル30. ライスパラメータ導出プロセス

このプロセスへの入力は、ベースレベルの`baseLevel`、カラー成分インデックス`cldx`、現在のピクチャの左上のサンプルを基準とした現在の変換ブロックの左上のサンプルを指定する輝度位置 $(x0, y0)$ 、現在の係数スキャン位置 (xC, yC) 、変換ブロック幅の2進対数 $\log_2 TbWidth$ 、および変換ブロックの高さの2進対数 $\log_2 TbHeight$ である。

このプロセスの出力は、ライスパラメータ`cRiceParam`である。

成分インデックス`cldx`と左上の輝度位置 $(x0, y0)$ を持つ変換ブロックの配列`AbsLevel[x][y]`を指定すると、変数`locSumAbs`は次の疑似コードで指定されているように導出される。

$Shift = \max(0, (BitDepth - 8)/2)$

`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 ]` (1494)

`}`

`if( yC < (1 << log2TbHeight) - 1 ) {`

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

`if( yC < (1 << log2TbHeight) - 2 )`

`locSumAbs += AbsLevel[ xC ][ yC + 2 ]`

`}`

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

変数`locSumAbs`を指定すると、テーブル4に示されるようにライスパラメータ`cRiceParam`が導出される。

$cRiceParam = cRiceParam + Shift$

`baseLevel`が0に等しい場合、変数`ZeroPos[n]`は次のように導出される。

$ZeroPos[n] = (QState < 2 ? 1 : 2) << cRiceParam$

【 0 2 5 0 】

【00256】<変換スキップのための残差コーディング>

【 0 2 5 1 】

【00257】本開示の第27の態様によれば、変換スキップの残差コーディングにおいて、特定のシンタックス要素、例えば`abs_remainder`を符号化するために2値符号語の変数セットを使用することが提案され、選択は現在のブロックの特定の符号化情報、例えば`TB/CB`および／またはスライス／プロファイルに関連付けられた量子化パラメータもしくは符号化ビット深度に従って、ならびに／あるいは`TB/CB/スライス/`

10

20

30

40

50

ピクチャ／シーケンスレベルに関連付けられた新しいフラグ、例えば、`extended_precision_processing_flag`に従って決定される。2値符号語の変数セットを導出するには、さまざまな方法を使用できる。いくつかの例示的な方法を以下に示す。

【0252】

[00258]まず、現在のVVCで使用されるのと、`abs_remainder`の符号語を決定するための同じ手順が使用されるが、常に固定ライスパラメータ（例えば、2、3、4、5、6、7または8）が選択される。固定値は、現在のブロックの特定の符号化情報、例えば、量子化パラメータ、フレームタイプ（例えば、I、PもしくはB）、成分ID（例えば、輝度もしくは色差）、カラーフォーマット（例えば、420、422、もしくは444）、またはTB／CBおよび／もしくはスライス／プロファイルに関連付けられた符号化ビット深度に従って、ならびに／あるいはTB／CB／スライス／ピクチャ／シーケンスレベルに関連付けられたシンタックス要素、例えば、`rice_parameter_value`に従って、異なる条件では異なり得る。具体例をテーブル7に示す。ここで、TH1～TH4は（TH1<TH2<TH3<TH4）を満たす所定の閾値であり、K0～K4は所定のライスパラメータである。実際には、同じロジックが異なる方法で実装され得ることは注目に値する。例えば、特定の方程式またはルックアップテーブルを使用して、テーブル7に示すように、現在のCU／シーケンスのBitDepth値から同じライスパラメータを導出することもできる。

10

【0253】

[00259]第2に、固定長の2値化。

【0254】

[00260]第3に、切り捨てライス2値化。

【0255】

[00261]第4に、切り捨て2値（TB）2値化プロセス。

【0256】

[00262]第5に、k次のExp-Golomb 2値化プロセス（EGk）。

【0257】

[00263]第6に、限定されたk次のExp-Golomb 2値化

【0258】

[00264]VVCドラフトに基づく対応する復号プロセスの例を以下に示す。VVCドラフトへの変更はテーブル31に太字および斜体フォントで示され、削除された内容は取り消し線フォントで示される。実際には、同じロジックが異なる方法で実装され得ることは注目に値する。例えば、特定の方程式またはルックアップテーブルを使用して、同じライスパラメータを導出することもできる。

30

【0259】

40

50

【表 3 1】

テーブル31. ライスパラメータ導出プロセス

ライスパラメータcRiceParamは次のように導出される。

- *transform_skip_flag[x0][y0][cldx]*が1に等しく、かつ*sh_ts_residual_coding_disabled_flag*が0の場合、ライスパラメータcRiceParamの導出プロセスは以下のよう指定される。

```

    if(BitDepth <11)
    {
        rice parameter = 1
    }
    else if(BitDepth <13)
    {
        rice parameter = 4
    }
    else if(BitDepth <15)
    {
        rice parameter = 6
    }
    else
    {
        rice parameter = 8
    }

```

- それ以外の場合、ライスパラメータcRiceParamは、変数baseLevelを4に設定して、カラー成分インデックスcldx、輝度位置(x0, y0)、現在の係数スキャン位置(xC, yC)、変換ブロック幅の2進対数log2TbWidth、および変換ブロック高さの2進対数log2TbHeightを入力としてテーブル3に指定されているabs_remainder[]のライスパラメータ導出プロセスを呼び出すことによって導出される。

【0 2 6 0】

[00265]別の例では、新しいフラグ、例えば*extended_precision_processing_flag*が1に等しい場合、abs_remainderのシンタックス要素を符号化する際にライスパラメータに対して1つの固定値のみを使用することが提案される。VVCドラフトに基づく対応する復号プロセスを以下に示す。変更は太字および斜体のフォントで、削除された内容は取り消し線のフォントで示されている。VVCドラフトへの変更は、テーブル32に太字および斜体のフォントで示されている。

【0 2 6 1】

10

20

30

40

50

【表 3 2】

テーブル32. ライスパラメータ導出プロセス

ライスパラメータcRiceParamは次のように導出される。

- *extended_precision_processing_flag*が1に等しく、*transform_skip_flag*[x0][y0][cldx]が1に等しく、*sh_ts_residual_coding_disabled_flag*が0に等しい場合、ライスパラメータcRiceParamの導出プロセスは以下のように規定される。

```

if(BitDepth <11)
{
    rice parameter = 1
}
else if(BitDepth <13)
{
    rice parameter = 4
}
else if(BitDepth <15)
{
    rice parameter = 6
}
else
{
    rice parameter = 8
}

```

- *transform_skip_flag*[x0][y0][cldx]が1に等しい場合、かつ *sh_ts_residual_coding_disabled_flag*は0に等しい場合、ライスパラメータcRiceParamは1に等しく設定される。

- それ以外の場合、ライスパラメータcRiceParamは、変数baseLevelを4に設定して、カラー成分インデックスcldx、輝度位置(x0, y0)、現在の係数スキャン位置(xC, yC)、変換ブロック幅の2進対数log2Tbwidth、および変換ブロック高さの2進対数log2TbHeightを入力としてテーブル3に指定されているabs_remainder[]のライスパラメータ導出プロセスを呼び出すことによって導出される。

【0 2 6 2】

[00266]さらに別の例では、新しいフラグ、例えば*extended_precision_processing_flag*が1に等しい場合、ライスパラメータcRiceParamはnに固定される。ここで、nは正の数（例えば、2、3、4、5、6、7、または8）である。条件により固定値が異なり得る。VVCドラフトに基づく対応する復号プロセスの例を以下に示す。変更は太字および斜体のフォントで示され、削除された内容は取り消し線のフォントで示されている。VVCドラフトへの変更は、テーブル33に太字および斜体のフォントで示されている。

【0 2 6 3】

10

20

30

40

50

【表 3 3】

テーブル33. ライスパラメータ導出プロセス

ライスパラメータcRiceParamは次のように導出される。

- *extended_precision_processing_flag*が1に等しく、*transform_skip_flag*[x0][y0][cldx]が1に等しく、*sh_ts_residual_coding_disabled_flag*が0に等しい場合、ライスパラメータcRiceParamは7に設定される。
- *transform_skip_flag*[x0][y0][cldx]が1に等しく、*sh_ts_residual_coding_disabled_flag*が0に等しい場合、ライスパラメータcRiceParamは1に等しく設定される。
- それ以外の場合、ライスパラメータcRiceParamは、変数baseLevelを4に設定して、カラー成分インデックスcldx、輝度位置(x0, y0)、現在の係数スキャン位置(xC, yC)、変換ブロック幅の2進対数log2Tbwidth、および変換ブロック高さの2進対数log2TbHeightを入力としてテーブル3に指定されているabs_remainder[]のライスパラメータ導出プロセスを呼び出すことによって導出される。

10

【0 2 6 4】

【00267】さらに別の例では、BitDepthが所定の閾値（例えば、10、11、12、13、14、15、または16）以上である場合、ライスパラメータcRiceParamはnに固定される。ここで、nは正の数、例えば、4、5、6、7、または8である。条件により固定値が異なり得る。VVCドラフトに基づく対応する復号プロセスの例を以下に示し、THは所定の閾値（例えば、10、11、12、13、14、15、または16）であり、変更は太字および斜体のフォントで示され、削除された内容は取り消し線のフォントで示される。VVCドラフトへの変更は、テーブル34に太字および斜体のフォントで示されている。

20

【0 2 6 5】

【表 3 4】

テーブル34. ライスパラメータ導出プロセス

ライスパラメータcRiceParamは次のように導出される。

- *BitDepth*がTHより大きく、*transform_skip_flag*[x0][y0][cldx]が1に等しく、*sh_ts_residual_coding_disabled_flag*が0に等しい場合、ライスパラメータcRiceParamは7に等しく設定される。
- *transform_skip_flag*[x0][y0][cldx]が1に等しい場合、かつ *sh_ts_residual_coding_disabled_flag*は0に等しい場合、ライスパラメータcRiceParamは1に等しく設定される。
- それ以外の場合、ライスパラメータcRiceParamは、変数baseLevelを4に設定して、カラー成分インデックスcldx、輝度位置(x0, y0)、現在の係数スキャン位置(xC, yC)、変換ブロック幅の2進対数log2Tbwidth、および変換ブロック高さの2進対数log2TbHeightを入力としてテーブル3に指定されているabs_remainder[]のライスパラメータ導出プロセスを呼び出すことによって導出される。

30

40

【0 2 6 6】

【00268】さらに別の例では、変換スキップブロックのライスパラメータの伝送が有効で

50

あるか無効であるかを示すために、1つの制御フラグがスライスヘッダ内で伝送される。制御フラグが有効として伝送されると、変換スキップスライスごとに1つのシンタックス要素がさらに伝送されて、そのスライスのライスパラメータが示される。制御フラグが無効として伝送されると（例えば、「0」に設定される）、変換スキップスライスのライスパラメータを示すために下位レベルでそれ以上のシンタックス要素は伝送されず、デフォルトのライスパラメータ（例えば、1）がすべての変換スキップスライスに使用される。VVCドラフトに基づく対応する復号プロセスの例を以下に示す。ここで、THは所定の値（例えば、0、1、2）であり、変更は太字および斜体のフォントで示され、削除された内容は取り消し線のフォントで示される。VVCドラフトへの変更は、テーブル35に太字および斜体のフォントで示されている。sh_ts_residual_coding_rice_indexはさまざまな方法で符号化でき、および／または最大値を有し得ることは注目に値する。例えば、u(n)、nビットを使用する符号なし整数、またはf(n)、左ビットを最初に（左から右に）書き込まれたnビットを使用する固定パターンビット文字列も、同じシンタックス要素の符号化／復号に使用できる。

【0267】

[00269]<スライスヘッダのシンタックス>

【0268】

【表35】

テーブル35. 残差コーディングのシンタックス

slice_header() {	記述子
...	
if(sps_transform_skip_enabled_flag && !sh_dep_quant_used_flag && !sh_sign_data_hiding_used_flag)	
sh_ts_residual_coding_disabled_flag	u(1)
<i>if(!sh_ts_residual_coding_disabled_flag) {</i>	
sh_ts_residual_coding_rice_flag	u(1)
<i>if(sh_ts_residual_coding_rice_flag)</i>	
sh_ts_residual_coding_rice_index	ue(v)
}	
...	

【0269】

[00270] 1に等しいsh_ts_residual_coding_rice_flagは、sh_ts_residual_coding_rice_indexが現在のスライスに存在し得ることを指定し、0に等しいsh_ts_residual_coding_rice_flagは、sh_ts_residual_coding_rice_indexが現在のスライスに存在しないことを指定する。sh_ts_residual_coding_rice_flagが存在しない場合、sh_ts_residual_coding_rice_flagの値は0に等しいと推定される。sh_ts_residual_coding_rice_indexは、residual_ts_coding()シンタックス構造に使用されるライスパラメータを指定する。

【0270】

【表36】

テーブル36. ライスパラメータ導出プロセス

ライスパラメータcRiceParamは次のように導出される。

- *sh_ts_residual_coding_rice_flag*が1に等しく、
transform_skip_flag[x0][y0][cldx]が1に等しく、
*sh_ts_residual_coding_disabled_flag*が0に等しい場合、ライスパラメータcRiceParamは(*sh_ts_residual_coding_rice_index*+TH)に等しく設定される。
- *transform_skip_flag*[x0][y0][cldx]が1に等しい場合、かつ
*sh_ts_residual_coding_disabled_flag*は0に等しい場合、ライスパラメータcRiceParamは1に等しく設定される。
- それ以外の場合、ライスパラメータcRiceParamは、変数baseLevelを4に設定して、カラー成分インデックスcldx、輝度位置(x0, y0)、現在の係数スキャン位置(xC, yC)、変換ブロック幅の2進対数log2Tbwidth、および変換ブロック高さの2進対数log2TbHeightを入力としてテーブル3に指定されているabs_remainder[]のライスパラメータ導出プロセスを呼び出すことによって導出される。

【0271】

【00271】さらに別の例では、1つの制御フラグがシーケンスパラメータセット（またはシーケンスパラメータセット範囲拡張シンタックス）内で伝送され、変換スキップブロックに対するライスパラメータの伝送が有効であるか無効であるかを示す。制御フラグが有効として伝送されると、変換スキップスライスごとに1つのシンタックス要素がさらに伝送されて、そのスライスのライスパラメータが示される。制御フラグが無効として伝送されると（例えば、「0」に設定される）、変換スキップスライスのライスパラメータを示すために下位レベルでそれ以上のシンタックス要素は伝送されず、デフォルトのライスパラメータ（例えば、1）がすべての変換スキップスライスに使用される。VVCドラフトに基づく対応する復号プロセスの例を以下に示す。ここで、THは、所定の値（例えば、0、1、2）である。VVCドラフトへの変更はテーブル37に太字および斜体のフォントで示され、削除された内容は取り消し線のフォントで示されている。*sh_ts_residual_coding_rice_index*はさまざまな方法で符号化できること、および／または最大値を有し得ることは注目に値する。例えば、 $u(n)$ 、 n ビットを使用する符号なし整数、または $f(n)$ 、左ビットを最初に（左から右に）書き込まれた n ビットを使用する固定パターンビット文字列も、同じシンタックス要素の符号化／復号に使用できる。

【0272】

【00272】<シーケンスパラメータセットRBSシンタックス>

【0273】

10

20

30

40

50

【表 3 7】

テーブル37. 残差コーディングのシンタックス

seq_parameter_set_rbsp() {	記述子
...	
sps_sign_data_hiding_enabled_flag	u(1)
<i>sps_ts_residual_coding_rice_present_in_sh_flag</i>	<i>u(1)</i>
sps_virtual_boundaries_enabled_flag	u(1)
...	
}	

10

【0 2 7 4】

[00273] 1に等しいsps_ts_residual_coding_rice_present_in_sh_flagは、SPSを参照するSHシンタックス構造内にsh_ts_residual_coding_rice_idxが存在し得ることを指定する。0に等しいsps_ts_residual_coding_rice_present_in_sh_flagは、SPSを参照するSHシンタックス構造内にsh_ts_residual_coding_rice_idxが存在しないことを指定する。sps_ts_residual_coding_rice_present_in_sh_flagが存在しない場合、sps_ts_residual_coding_rice_present_in_sh_flagの値は0に等しいと推定される。

20

【0 2 7 5】

[00274] <スライスヘッダのシンタックス>

【0 2 7 6】

【表 3 8】

30

テーブル38. 残差コーディングのシンタックス

slice_header() {	記述子
...	
if(sps_transform_skip_enabled_flag && !sh_dep_quant_used_flag && !sh_sign_data_hiding_used_flag)	
sh_ts_residual_coding_disabled_flag	u(1)
<i>if(!sh_ts_residual_coding_disabled_flag) && sps_ts_residual_coding_rice_enabled_flag)</i>	
<i>sh_ts_residual_coding_rice_idx</i>	<i>ue(v)</i>
...	
}	

40

50

【0277】

[00275] `sh_ts_residual_coding_rice_idx`は、`residual_ts_coding()` シンタックス構造に使用されるライスパラメータを指定する。

【0278】

【表39】

テーブル39. ライスパラメータ導出プロセス

ライスパラメータ`cRiceParam`は次のように導出される。

- `sps_ts_residual_coding_rice_flag`が1に等しく、
`transform_skip_flag[x0][y0][cldx]`が1に等しく、
`sh_ts_residual_coding_disabled_flag`が0に等しい場合、ライスパラメータ`cRiceParam`は $(sh_ts_residual_coding_rice_idx + TH)$ に等しく設定される。
- `transform_skip_flag[x0][y0][cldx]`が1に等しく、`sh_ts_residual_coding_disabled_flag`が0に等しい場合、ライスパラメータ`cRiceParam`は1に等しく設定される。
- それ以外の場合、ライスパラメータ`cRiceParam`は、変数`baseLevel`を4に設定して、カラー成分インデックス`cldx`、輝度位置(`x0`, `y0`)、現在の係数スキャン位置(`xC`, `yC`)、変換ブロック幅の2進対数`log2Tbwidth`、および変換ブロック高さの2進対数`log2TbHeight`を入力としてテーブル3に指定されている`abs_remainder[]`のライスパラメータ導出プロセスを呼び出すことによって導出される。

【0279】

[00276]さらに別の例では、各変換スキップスライスに対して1つのシンタックス要素が伝送されて、そのスライスのライスパラメータを示す。VVCドラフトに基づく対応する復号プロセスの例を以下に示す。VVCドラフトへの変更は、テーブル40に太字および斜体のフォントで示されている。`sh_ts_residual_coding_rice_idx`はさまざまな方法で符号化できること、および／または最大値を有し得ることは注目に値する。例えば、`u(n)`、`n`ビットを使用する符号なし整数、または`f(n)`、左ビットを最初に（左から右に）書き込まれた`n`ビットを使用する固定パターンビット文字列も、同じシンタックス要素の符号化／復号に使用できる。

【0280】

[00277] <スライスヘッダのシンタックス>

【0281】

10

20

30

40

50

【表 4 0】

テーブル40. 残差コーディングのシンタックス

slice_header() {	記述子
...	
if(sps_transform_skip_enabled_flag && !sh_dep_quant_used_flag && !sh_sign_data_hiding_used_flag)	
sh_ts_residual_coding_disabled_flag	u(1)
<i>if(!sh_ts_residual_coding_disabled_flag)</i>	
<i>sh_ts_residual_coding_rice_idx</i>	<i>ue(v)</i>
...	
}	

10

【 0 2 8 2】

20

[00278] `sh_ts_residual_coding_rice_idx`は、`residual_ts_coding()` シンタックス構造に使用されるライスパラメータを指定する。`sh_ts_residual_coding_rice_idx`が存在しない場合、`sh_ts_residual_coding_rice_idx`の値は0に等しいと推定される。

【 0 2 8 3】

【表 4 1】

テーブル41. ライスパラメータ導出プロセス

ライスパラメータcRiceParamは次のように導出される。
— <code>transform_skip_flag[x0][y0][cldx]</code> が1に等しい場合、かつ <code>sh_ts_residual_coding_disabled_flag</code> は0に等しい場合、ライスパラメータcRiceParamは <code>sh_ts_residual_coding_rice_idx+1</code> に等しく設定される。
— それ以外の場合、ライスパラメータcRiceParamは、変数baseLevelを4に設定して、カラー成分インデックスcldx、輝度位置(x0, y0)、現在の係数スキャン位置(xC, yC)、変換ブロック幅の2進対数log2Tbwidth、および変換ブロック高さの2進対数log2TbHeightを入力としてテーブル3に指定されているabs_remainder[]のライスパラメータ導出プロセスを呼び出すことによって導出される。

30

40

【 0 2 8 4】

[00279] さらに別の例では、変換スキップブロックのライスパラメータの伝送が有効か無効かを示すために、ピクチャパラメータセット範囲拡張シンタックスで1つの制御フラグが伝送される。制御フラグが有効として伝送されると、そのピクチャのライスパラメータを示すために1つのシンタックス要素がさらに伝送される。制御フラグが無効として伝送されると（例えば、「0」に設定される）、変換スキップスライスのライスパラメータを示すために下位レベルでそれ以上のシンタックス要素は伝送されず、デフォルトのライスパラメータ（例えば、1）がすべての変換スキップスライスに使用される。VVCドラフトに基づく対応する復号プロセスの例を以下に示す。ここで、THは、所定の値（例え

50

ば、0、1、2）である。VVCドラフトへの変更は、テーブル42に太字および斜体のフォントで示されている。`pps_ts_residual_coding_rice_idx`はさまざまな方法で符号化できること、および／または最大値を有し得ることは注目に値する。例えば、 $u(n)$ 、 n ビットを使用する符号なし整数、または $f(n)$ 、左ビットを最初に（左から右に）書き込まれた n ビットを使用する固定パターンビット文字列も、同じシンタックス要素の符号化／復号に使用できる。

【0285】

[00280]＜ピクチャパラメータセット範囲の拡張シンタックス＞

【0286】

【表42】

10

テーブル42. 残差コーディングのシンタックス

<code>pps_range_extensions() {</code>	記述子
<code>...</code>	
<i>pps_ts_residual_coding_rice_flag</i>	<i>u(1)</i>
<i>if(pps_ts_residual_coding_rice_flag)</i>	
<i>pps_ts_residual_coding_rice_idx</i>	<i>ue(v)</i>
<code>}</code>	
<code>...</code>	

20

【0287】

[00281]1に等しい`pps_ts_residual_coding_rice_flag`は、現在のピクチャに`pps_ts_residual_coding_rice_index`が存在し得ることを指定し、0に等しい`pps_ts_residual_coding_rice_flag`は、`pps_ts_residual_coding_rice_idx`が現在のピクチャに存在しないことを指定する。`pps_ts_residual_coding_rice_flag`が存在しない場合、`pps_ts_residual_coding_rice_flag`の値は0に等しいと推定される。

30

【0288】

[00282]`pps_ts_residual_coding_rice_idx`は、`residual_ts_coding()`シンタックス構造に使用されるライスパラメータを指定する。

【0289】

40

50

【表 4 3】

テーブル43. ライスパラメータ導出プロセス

ライスパラメータcRiceParamは次のように導出される。

- *pps_ts_residual_coding_rice_flag*が1に等しく、
transform_skip_flag[x0][y0][cldx]が1に等しく、
*sh_ts_residual_coding_disabled_flag*が0に等しい場合、ライスパラメータcRiceParamは(*pps_ts_residual_coding_rice_idx*+TH)に等しく設定される。
- *transform_skip_flag*[x0][y0][cldx]が1に等しい場合、かつ
*sh_ts_residual_coding_disabled_flag*は0に等しい場合、ライスパラメータcRiceParamは1に等しく設定される。
- それ以外の場合、ライスパラメータcRiceParamは、変数baseLevelを4に設定して、カラー成分インデックスcldx、輝度位置(x0、y0)、現在の係数スキャン位置(xC、yC)、変換ブロック幅の2進対数log2Tbwidth、および変換ブロック高さの2進対数log2TbHeightを入力としてテーブル3に指定されているabs_remainder[]のライスパラメータ導出プロセスを呼び出すことによって導出される。

【0 2 9 0】

[00283]さらに別の例では、シンタックス要素abs_remainderの符号化に可変ライスパラメータのみを使用することが提案される。適用されるライスパラメータの値は、現在のブロックの特定の符号化情報、例えば、ブロックサイズ、量子化パラメータ、ビット深度、変換タイプなどに従って決定され得る。1つの特定の実施形態では、符号化ビット深度および1つのCUに適用される量子化パラメータに基づいてライスパラメータを調整することが提案される。VVCドラフトに基づく対応する復号プロセスを以下に示す。VVCドラフトへの変更はテーブル44に太字および斜体フォントで示され、削除された内容は取り消し線フォントで示される。実際には、同じロジックが異なる方法で実装され得ることは注目に値する。例えば、特定の方程式またはルックアップテーブルを使用して、同じライスパラメータを導出することもできる。

【0 2 9 1】

10

20

30

40

50

【表 4 4 - 1】

テーブル44. ライスパラメータ導出プロセス

ライスパラメータcRiceParamは次のように導出される。

- *transform_skip_flag*[x0][y0][cldx]が1に等しい場合、かつ
*sh_ts_residual_coding_disabled_flag*が0に等しい場合、ライスパラメータcRiceParamの導出プロセスは以下のように指定される。

```
if(BitDepth <11)
{
    rice parameter = 1
}
else if(BitDepth <13)
{
    if(QPCU <-15)
    {
        rice parameter = 6
    }
    else if(QPCU <-10)
    {
        rice parameter = 5
    }
    else if(QPCU <0)
    {
        rice parameter = 4
    }
    else if(QPCU <10)
    {
        rice parameter = 3
    }
    else
    {
        rice parameter = 2
    }
}
```

10

20

30

40

50

【表 4 4 - 2 】

<pre> } else if(BitDepth <15) { if(QP_{CU} <-25) { rice parameter = 7 } else if(QP_{CU} <-15) { rice parameter = 6 } else if(QP_{CU} <-10) { rice parameter = 5 } else { rice parameter = 4 } } else { if(QP_{CU} <-30) { rice parameter = 8 } else if(QP_{CU} <-25) { rice parameter = 7 } } </pre>	
--	--

10

20

30

40

50

【表 4 4 - 3】

```

else if( $QP_{CU} < -15$ )
{
    rice parameter = 6
}
else if( $QP_{CU} < -10$ )
{
    rice parameter = 5
}
else
{
    rice parameter = 4
}
}

```

— それ以外の場合、ライスパラメータ $cRiceParamI$ は、変数 $baseLevel$ を 4 に設定して、カラー成分インデックス $cIdx$ 、輝度位置 $(x0, y0)$ 、現在の係数スキャン位置 (xC, yC) 、変換ブロック幅の2進対数 $\log_2 Tbwidth$ 、および変換ブロック高さの2進対数 $\log_2 TbHeight$ を入力としてテーブル 3 に指定されている $abs_remainder[]$ のライスパラメータ導出プロセスを呼び出すことによって導出される。

【0 2 9 2】

[00284] さらに別の例では、VVCドラフトに基づく対応する復号プロセスが以下のよう
に示され、ここで、THは、所定の閾値（例えば、33または34）である。VVCドラフト
への変更はテーブル 45 に太字および斜体のフォントで示され、削除された内容は
取り消し線のフォントで示されている。実際には、同じロジックが異なる方法で実装され
得ることは注目に値する。例えば、特定の方程式またはルックアップテーブルを使用して
、同じライスパラメータを導出することもできる。

【0 2 9 3】

10

20

30

40

50

【表 4 5】

テーブル45. ライスパラメータ導出プロセス

ライスパラメータcRiceParamは次のように導出される。

- *transform_skip_flag[x0][y0][cldx]*が1に等しい場合、かつ
*sh_ts_residual_coding_disabled_flag*が0に等しい場合、ライスパラメータcRiceP
aramの導出プロセスは以下のように指定される。
ライスパラメータ= $\text{Clip3}(1, 8, \text{Floor}((TH - \text{BitDepth} - QP_{cu}) / 6))$
- それ以外の場合、ライスパラメータcRiceParamは、変数baseLevelを4に設定して、カラー
成分インデックスcldx、輝度位置(x0, y0)、現在の係数スキャン位置(xC, yC)、変換ブロッ
ク幅の2進対数log2Tbwidth、および変換ブロック高さの2進対数log2TbHeightを入力とし
てテーブル3に指定されているabs_remainder[]のライスパラメータ導出プロセスを呼び
出すことによって導出される。

10

【0 2 9 4】

[00285]さらに別の例では、VVCドラフトに基づく対応する復号プロセスが以下のよ
うに示され、THAおよびTHBは所定の閾値（例えば、THA=8、THB=33または
34）である。VVCドラフトへの変更はテーブル46に太字および斜体のフォントで示
され、削除された内容は取り消し線のフォントで示されている。実際には、同じロジック
が異なる方法で実装され得ることは注目に値する。例えば、特定の方程式またはルックア
ップテーブルを使用して、同じライスパラメータを導出することもできる。

20

【0 2 9 5】

【表 4 6】

テーブル46. ライスパラメータ導出プロセス

ライスパラメータcRiceParamは次のように導出される。

- *transform_skip_flag[x0][y0][cldx]*が1に等しい場合、かつ
*sh_ts_residual_coding_disabled_flag*が0に等しい場合、ライスパラメータcRiceP
aramの導出プロセスは以下のように指定される。
ライスパラメータ= $\text{Clip3}(1, \text{BitDepth} - TH_A, \text{Floor}((TH_B - \text{BitDepth} - QP_{cu}) / 6))$
- それ以外の場合、ライスパラメータcRiceParamは、変数baseLevelを4に設定して、カラー
成分インデックスcldx、輝度位置(x0, y0)、現在の係数スキャン位置(xC, yC)、変換ブロッ
ク幅の2進対数log2Tbwidth、および変換ブロック高さの2進対数log2TbHeightを入力とし
てテーブル3に指定されているabs_remainder[]のライスパラメータ導出プロセスを呼び
出すことによって導出される。

30

40

【0 2 9 6】

[00286]さらに別の例では、新しいフラグ、例えばextended_precision
on_processing_flagが1に等しい場合、abs_remainder
のシンタックス要素の符号化に可変ライスパラメータのみを使用することが提案される。
変動する値は、現在のブロックの特定の符号化情報、例えば、ブロックサイズ、量子化パ
ラメータ、ビット深度、変換タイプなどに従って決定され得る。1つの特定の実施形態で
は、符号化ビット深度および1つのCUに適用される量子化パラメータに基づいてライス
パラメータを調整することが提案される。VVCドラフトに基づく対応する復号プロセス
を以下に示す。VVCドラフトへの変更は、テーブル47に太字および斜体のフォントで

50

示されている。実際には、同じロジックが異なる方法で実装され得ることは注目に値する。例えば、特定の方程式またはルックアップテーブルを使用して、同じライスパラメータを導出することもできる。

【 0 2 9 7 】

【表 4 7 - 1 】

テーブル47. ライスパラメータ導出プロセス

ライスパラメータ $cRiceParam$ は次のように導出される。

— $extended_precision_processing_flag$ が 1 に等しく、
 $transform_skip_flag[x0][y0][cldx]$ が 1 に等しく、
 $sh_ts_residual_coding_disabled_flag$ が 0 に等しい場合、ライスパラメータ $cRiceParam$ の導出プロセスは以下のように規定される。

```

    if( $QP_{CU} \leq -30$ )
    {
        rice parameter = 8
    }
    else if( $QP_{CU} \leq -25$ )
    {
        rice parameter = 7
    }
    else if( $QP_{CU} \leq -15$ )
    {
        rice parameter = 6
    }
    else if( $QP_{CU} \leq -10$ )
    {
        rice parameter = 5
    }
    else if( $QP_{CU} < 0$ )
    {
        rice parameter = 4
    }
    else if( $QP_{CU} < 10$ )
    {
        rice parameter = 3
    }
    else if( $QP_{CU} < 15$ )

```

10

20

30

40

50

【表 4 7 - 2】

<pre> { rice parameter = 2 } else { rice parameter = 1 } </pre>	10
— transform__skip__flag[x0][y0][cldx]が1に等しく、sh_ts_residual_coding_disabled_flagが0に等しい場合、ライスパラメータcRiceParamは1に等しく設定される。 — それ以外の場合、ライスパラメータcRiceParamは、変数baseLevelを4に設定して、カラー成分インデックスcldx、輝度位置(x0, y0)、現在の係数スキャン位置(xC, yC)、変換ブロック幅の2進対数log2Tbwidth、および変換ブロック高さの2進対数log2TbHeightを入力としてテーブル3に指定されているabs__remainder[]のライスパラメータ導出プロセスを呼び出すことによって導出される。	20

【0 2 9 8】

【00287】さらに別の例では、VVCドラフトに基づく対応する復号プロセスが以下のよう
に示され、ここで、THは、所定の閾値（例えば、18、19）である。VVCドラフト
への変更は、テーブル48に太字および斜体のフォントで示されている。実際には、同
じロジックが異なる方法で実装され得ることは注目に値する。例えば、特定の方程式また
はルックアップテーブルを使用して、同じライスパラメータを導出することもできる。

【0 2 9 9】

【表 4 8】

テーブル48. ライスパラメータ導出プロセス

ライスパラメータcRiceParamは次のように導出される。 — <i>extended_precision_processing_flag</i>が1に等しい場合、 <i>transform_skip_flag</i>[x0][y0][cldx]が1に等しく、 <i>sh_ts_residual_coding_disabled_flag</i>が0に等しい場合、ライスパラメータcRiceParamの導出プロセスは次のように指定される。 ライスパラメータ=Clip3(1, 8, (TH-QP)/6) — transform__skip__flag[x0][y0][cldx]が1に等しく、sh_ts_residual_coding_disabled_flagが0に等しい場合、ライスパラメータcRiceParamは1に等しく設定される。 — それ以外の場合、ライスパラメータcRiceParamは、変数baseLevelを4に設定して、カラー成分インデックスcldx、輝度位置(x0, y0)、現在の係数スキャン位置(xC, yC)、変換ブロック幅の2進対数log2Tbwidth、および変換ブロック高さの2進対数log2TbHeightを入力としてテーブル3に指定されているabs__remainder[]のライスパラメータ導出プロセスを呼び出すことによって導出される。	40
---	----

【0 3 0 0】

10

20

30

40

50

[00288]さらに別の例では、VVCドラフトに基づく対応する復号プロセスが以下のよう
に示され、 TH_A および TH_B は所定の閾値（例えば、 $TH_A = 8$ 、 $TH_B = 18$ または
19）である。VVCドラフトへの変更は、テーブル49に太字および斜体のフォントで
示されている。実際には、同じロジックが異なる方法で実装され得ることは注目に値する
。例えば、特定の方程式またはルックアップテーブルを使用して、同じライスパラメータ
を導出することもできる。

【0301】

【表49】

テーブル49. ライスパラメータ導出プロセス

ライスパラメータ $cRiceParam$ は次のように導出される。

— *extended_precision_processing_flag*が1に等しく、
transform_skip_flag[$x0$][$y0$][$cldx$]が1に等しく、
*sh_ts_residual_coding_disabled_flag*が0に等しい場合、ライスパラメータ $cRiceP$
*aram*の導出プロセスは以下のように指定される。

$$\text{ライスパラメータ} = \text{Clip3}(1, \text{BitDepth} - TH_A, (TH_B - QP) / 6)$$

— *transform_skip_flag*[$x0$][$y0$][$cldx$]が1に等しく、*sh_ts_residual_coding_disa*
*bled_flag*が0に等しい場合、ライスパラメータ $cRiceParam$ は1に等しく設定される。

— それ以外の場合、ライスパラメータ $cRiceParam$ は、変数 $baseLevel$ を4に設定して、カラー
成分インデックス $cldx$ 、輝度位置($x0$, $y0$)、現在の係数スキャン位置(xC , yC)、変換ブロッ
ク幅の2進対数 $\log_2 Tbwidht$ 、および変換ブロック高さの2進対数 $\log_2 TbHeight$ を入力とし
てテーブル3に指定されている $abs_remainder[]$ のライスパラメータ導出プロセスを呼び
出すことによって導出される。

【0302】

[00289]図16はビデオ符号化の方法を示す。この方法は、例えばエンコーダに適用す
ることができる。

【0303】

[00290]ステップ1610で、エンコーダはビデオ入力を受信することができる。例え
ば、ビデオ入力はライブストリームであり得る。

【0304】

[00291]ステップ1612で、エンコーダはビデオ入力に基づいて量子化パラメータを
取得することができる。量子化パラメータは、例えば、エンコーダ内の量子化ユニットに
よって計算され得る。

【0305】

[00292]ステップ1614で、エンコーダは、少なくとも1つの所定の閾値、符号化ビ
ット深度、および量子化パラメータに基づいてライスパラメータを導出することができる
。例えば、ライスパラメータは、 $abs_remainder$ と dec_abs_level
のシンタックスを伝送するために使用される。

【0306】

[00293]ステップ1616で、エンコーダは、ライスパラメータに基づいてビデオビ
ットストリームをエントロピー符号化することができる。例えば、ビデオビットストリーム
は、圧縮されたビデオビットストリームを生成するためにエントロピー符号化されてもよ
い。

【0307】

[00294]さらに別の例では、 $BitDepth$ が10より大きい場合、 abs_rem
 $ainder$ のシンタックス要素を符号化する際に、ライスパラメータに固定値（例えば

、2、3、4、5、6、7、または8)のみを使用することが提案されている。固定値は、現在のブロックの特定の符号化情報、例えば量子化パラメータに従って、異なる条件では異なり得る。VVCドラフトに基づく対応する復号プロセスは以下のように示される。ここで、THは、所定の閾値（例えば、18、19）である。VVCドラフトへの変更は、テーブル50に太字および斜体のフォントで示されている。実際には、同じロジックが異なる方法で実装され得ることは注目に値する。例えば、特定の方程式またはルックアップテーブルを使用して、同じライスパラメータを導出することもできる。

【0308】

【表50】

テーブル50. ライスパラメータ導出プロセス

ライスパラメータcRiceParamは次のように導出される。

— *BitDepth*が10より大きく、*transform_skip_flag*[x0][y0][cldx]が1に等しく、*sh_ts_residual_coding_disabled_flag*が0に等しい場合、ライスパラメータcRiceParamの導出プロセスは次のように指定される。

$$\text{ライスパラメータ} = \text{Clip3}(1, 8, (TH - QP) / 6)$$

— *transform_skip_flag*[x0][y0][cldx]が1に等しく、*sh_ts_residual_coding_disabled_flag*が0に等しい場合、ライスパラメータcRiceParamは1に等しく設定される。

— それ以外の場合、ライスパラメータcRiceParamは、変数baseLevelを4に設定して、カラー成分インデックスcldx、輝度位置(x0, y0)、現在の係数スキャン位置(xC, yC)、変換ブロック幅の2進対数log2Tbwidth、および変換ブロック高さの2進対数log2TbHeightを入力としてテーブル3に指定されているabs_remainder[]のライスパラメータ導出プロセスを呼び出すことによって導出される。

【0309】

[00295]さらに別の例では、VVCドラフトに基づく対応する復号プロセスが以下のように示され、TH_AおよびTH_Bは所定の閾値（例えば、TH_A=8、TH_B=18または19）である。VVCドラフトへの変更は、テーブル51に太字および斜体のフォントで示されている。実際には、同じロジックが異なる方法で実装され得ることは注目に値する。例えば、特定の方程式またはルックアップテーブルを使用して、同じライスパラメータを導出することもできる。

【0310】

10

20

30

40

50

【表 5 1】

テーブル51. ライスパラメータ導出プロセス

ライスパラメータcRiceParamは次のように導出される。

- *BitDepth*が10より大きく、*transform_skip_flag*[*x0*][*y0*][*cldx*]が1に等しく、*sh_ts_residual_coding_disabled_flag*が0に等しい場合、ライスパラメータcRiceParamの導出プロセスは次のように指定される。

$$\text{ライスパラメータ} = \text{Clip3}(1, \text{BitDepth} - TH_A, (TH_B - QP) / 6)$$
- *transform_skip_flag*[*x0*][*y0*][*cldx*]が1に等しく、*sh_ts_residual_coding_disabled_flag*が0に等しい場合、ライスパラメータcRiceParamは1に等しく設定される。
- それ以外の場合、ライスパラメータcRiceParamは、変数baseLevelを4に設定して、カラー成分インデックス*cldx*、輝度位置(*x0*, *y0*)、現在の係数スキャン位置(*xC*, *yC*)、変換ブロック幅の2進対数log2Tbwidth、および変換ブロック高さの2進対数log2TbHeightを入力としてテーブル3に指定されているabs_remainder[]のライスパラメータ導出プロセスを呼び出すことによって導出される。

10

20

【0 3 1 1】

[00296]さらに別の例では、VVCドラフトに基づく対応する復号プロセスが以下のように示され、ここで、THは、所定の閾値（例えば、33または34）である。VVCドラフトへの変更は、テーブル52に太字と斜体のフォントで示されている。実際には、同じロジックが異なる方法で実装され得ることは注目に値する。例えば、特定の方程式またはルックアップテーブルを使用して、同じライスパラメータを導出することもできる。

【0 3 1 2】

【表 5 2】

テーブル52. ライスパラメータ導出プロセス

ライスパラメータcRiceParamは次のように導出される。

- *BitDepth*が10より大きく、*transform_skip_flag*[*x0*][*y0*][*cldx*]が1に等しく、*sh_ts_residual_coding_disabled_flag*が0に等しい場合、ライスパラメータcRiceParamの導出プロセスは次のように指定される。

$$\text{ライスパラメータ} = \text{Clip3}(1, 8, (TH - \text{BitDepth} - QP_{cu}) / 6)$$
- *transform_skip_flag*[*x0*][*y0*][*cldx*]が1に等しく、*sh_ts_residual_coding_disabled_flag*が0に等しい場合、ライスパラメータcRiceParamは1に等しく設定される。
- それ以外の場合、ライスパラメータcRiceParamは、変数baseLevelを4に設定して、カラー成分インデックス*cldx*、輝度位置(*x0*, *y0*)、現在の係数スキャン位置(*xC*, *yC*)、変換ブロック幅の2進対数log2Tbwidth、および変換ブロック高さの2進対数log2TbHeightを入力としてテーブル3に指定されているabs_remainder[]のライスパラメータ導出プロセスを呼び出すことによって導出される。

30

40

【0 3 1 3】

[00297]さらに別の例では、VVCドラフトに基づく対応する復号プロセスが以下のように示され、TH_AおよびTH_Bは所定の閾値（例えば、TH_A = 8、TH_B = 33または

50

34) である。VVCドラフトへの変更は、テーブル53に太字および斜体のフォントで示されている。実際には、同じロジックが異なる方法で実装され得ることは注目に値する。例えば、特定の方程式またはルックアップテーブルを使用して、同じライスパラメータを導出することもできる。

【0314】

【表53】

テーブル53. ライスパラメータ導出プロセス

ライスパラメータcRiceParamは次のように導出される。

- *BitDepth*が10より大きく、*transform_skip_flag*[x0][y0][cldx]が1に等しく、*sh_ts_residual_coding_disabled_flag*が0に等しい場合、ライスパラメータcRiceParamの導出プロセスは次のように指定される。

$$\text{ライスパラメータ} = \text{Clip3}(1, \text{BitDepth} - TH_A, (\text{TH}_B - \text{BitDepth} - QP_{cu}) / 6)$$
- *transform_skip_flag*[x0][y0][cldx]が1に等しく、*sh_ts_residual_coding_disabled_flag*が0に等しい場合、ライスパラメータcRiceParamは1に等しく設定される。
- それ以外の場合、ライスパラメータcRiceParamは、変数baseLevelを4に設定して、カラー成分インデックスcldx、輝度位置(x0, y0)、現在の係数スキャン位置(xC, yC)、変換ブロック幅の2進対数log2Tbwidth、および変換ブロック高さの2進対数log2TbHeightを入力としてテーブル3に指定されているabs_remainder[]のライスパラメータ導出プロセスを呼び出すことによって導出される。

【0315】

【00298】上記の図では、特定のライスパラメータを計算するために使用される方程式は、提案されたアイデアを説明するための例としてのみ使用されることに言及する価値がある。現代のビデオコーディング技術の当業者にとって、他のマッピング関数（または同等のマッピング方程式）は、提案されたアイデア（すなわち、符号化ビットおよび適用される量子化パラメータに基づいて変換スキップモードのライスパラメータを決定する）に既に適用可能である。一方、現在のVVC設計では、適用される量子化パラメータの値が符号化ブロックグループレベルで変更できることに言及する必要がある。したがって、提案されたライスパラメータ調整方式は、符号化ブロックグループレベルでの変換スキップモードのライスパラメータの柔軟な適応を提供することができる。

【0316】

【00299】＜正規の残差コーディングおよび変換スキップ残差コーディングのための伝送情報＞

【0317】

【00300】本開示の第28の態様によれば、正規の残差コーディングにおいて、特定のシンタックス要素、例えば変換スキップ残差コーディングのabs_remainder、abs_remainder / dec_abs_levelに使用されるライスパラメータを導出するためのシフトパラメータとオフセットパラメータを符号化するために2値符号語のライスパラメータを伝送することと、現在のブロックの特定の符号化情報、例えばTB / CBおよび／またはスライス / プロファイルに関連付けられた量子化パラメータもしくは符号化ビット深度に従って、ならびに／あるいはTB / CB / スライス / ピクチャ / シーケンスレベルに関連付けられた新しいフラグ、例えば、sps_residual_coding_info_present_in_sh_flagに従って伝送するかどうかを決定することとが提案されている。

【0318】

[00301]一例では、1つの制御フラグは、変換スキップブロックのライスパラメータの伝送と、変換ブロックのライスパラメータ導出用のシフトパラメータおよび／またはオフセットパラメータの伝送が有効か無効かを示すために、スライスヘッダで伝送される。制御フラグが有効であると伝送されると、1つのシンタックス要素は、各変換スキップスライスに対してさらに伝送されて、そのスライスのライスパラメータを示し、2つのシンタックス要素は、各変換スライスに対してさらに伝送されて、そのスライスのライスパラメータを導出するためのシフトパラメータおよび／またはオフセットパラメータを示す。制御フラグが無効として伝送されると（例えば、「0」に設定される）、変換スキップスライスのライスパラメータを示すために下位レベルでそれ以上のシンタックス要素は伝送されず、デフォルトのライスパラメータ（例えば、1）がすべての変換スキップスライスに使用され、変換スライスのライスパラメータ導出のためのシフトパラメータおよびオフセットパラメータを示すために、下位レベルでそれ以上のシンタックス要素は信号送信されず、デフォルトのシフトパラメータおよび／またはオフセットパラメータ（例えば、0）がすべての変換スライスに使用される。VVCドラフトに基づく対応する復号プロセスの例を以下に示す。ここで、THは、所定の値（例えば、0、1、2）である。VVCドラフトへの変更は、テーブル54に太字および斜体のフォントで示されている。sh_residual_coding_rice_shift、sh_residual_coding_rice_offset、およびsh_ts_residual_coding_rice_indexは、異なる方法で符号化することができ、および／または最大値を有することができることは注目に値する。例えば、u(n)、nビットを使用する符号なし整数、またはf(n)、左ビットを最初に（左から右に）書き込まれたnビットを使用する固定パターンビット文字列も使用でき、同じシンタックス要素を符号化／復号する。

[0319]

[00302]図17は、ビデオ復号の方法を示す。この方法は、例えばエンコーダに適用することができる。

[0320]

[00303]ステップ1710で、エンコーダはビデオ入力を受信することができる。

[0321]

[00304]ステップ1712で、エンコーダは、シンタックス要素を符号化するための2値符号語のライスパラメータを伝送することができる。符号化シンタックス要素には、変換スキップ残差コーディングのabs_remainderが含まれ得る。

[0322]

[00305]ステップ1714で、エンコーダは、ライスパラメータおよびビデオ入力に基づいてビデオビットストリームをエントロピー符号化することができる。

[0323]

[00306]<スライスヘッダのシンタックス>

[0324]

10

20

30

40

50

【表 5 4】

テーブル54. 残差コーディングのシンタックス

slice_header() {	記述子
...	
if(sps_transform_skip_enabled_flag && !sh_dep_quant_used_flag && !sh_sign_data_hiding_used_flag)	
sh_ts_residual_coding_disabled_flag	u(1)
<i>sh_residual_coding_rice_flag</i>	<i>u(1)</i>
<i>if(sh_ts_residual_coding_rice_flag) {</i>	
<i>sh_residual_coding_rice_shift</i>	<i>ue(v)</i>
<i>sh_residual_coding_rice_offset</i>	<i>ue(v)</i>
<i>if(!sh_ts_residual_coding_disabled_flag)</i>	
<i>sh_ts_residual_coding_rice_index</i>	<i>ue(v)</i>
}	
...	

【0325】

[00307] 1に等しい `sh_residual_coding_rice_flag` は、`sh_residual_coding_rice_shift`、`sh_residual_coding_rice_offset`、`sh_residual_coding_rice_index` が現在のスライスに存在し得ることを指定し、0に等しい `sh_residual_coding_rice_flag` は、`sh_residual_coding_rice_shift`、`sh_residual_coding_rice_offset`、`sh_residual_coding_rice_index` が現在のスライスに存在しないことを指定する。

【0326】

[00308] `sh_residual_coding_rice_shift` は、`abs_remainder[]` および `dec_abs_level[]` のライスパラメータ導出プロセスに使用されるシフトパラメータを指定する。`sh_residual_coding_rice_shift` が存在しない場合、`sh_residual_coding_rice_shift` の値は0に等しいと推定される。

【0327】

[00309] `sh_residual_coding_rice_offset` は、`abs_remainder[]` および `dec_abs_level[]` のライスパラメータ導出プロセスに使用されるオフセットパラメータを指定する。`sh_residual_coding_rice_offset` が存在しない場合、`sh_residual_coding_rice_offset` の値は0に等しいと推定される。

【0328】

[00310] `sh_ts_residual_coding_rice_index` は、`residual_ts_coding()` シンタックス構造に使用されるライスパラメータを指定する。`sh_ts_residual_coding_rice_index` が

存在しない場合、`sh_ts_residual_coding_rice_index`の値は0に等しいと推定される。

【0329】

【表55】

テーブル55. ライスパラメータ導出プロセス

`abs_remainder[ ]`の2値化プロセス

...

ライスパラメータ`cRiceParam`は次のように導出される。

- `sh_residual_coding_rice_flag`が1に等しく、`transform_skip_flag[x0][y0][cldx]`が1に等しく、`sh_ts_residual_coding_disabled_flag`が0に等しい場合、ライスパラメータ`cRiceParam`は $(sh_ts_residual_coding_rice_index + TH)$ に等しく設定される。
- `transform_skip_flag[x0][y0][cldx]`が1に等しく、`sh_ts_residual_coding_disabled_flag`が0に等しい場合、ライスパラメータ`cRiceParam`は1に等しく設定される。
- それ以外の場合、ライスパラメータ`cRiceParam`は、変数`baseLevel`を4に設定して、カラー成分インデックス`cldx`、輝度位置(`x0`, `y0`)、現在の係数スキャン位置(`xC`, `yC`)、変換ブロック幅の2進対数 $\log_2 Tbwidth$ 、および変換ブロック高さの2進対数 $\log_2 TbHeight$ を入力としてテーブル3に指定されている`abs_remainder[ ]`のライスパラメータ導出プロセスを呼び出すことによって導出される。

【0330】

10

20

30

40

50

【表 5 6】

テーブル56. ライスパラメータ導出プロセス

abs__remainder[]とdec__abs__level[]のライスパラメータ導出プロセス

このプロセスへの入力は、ベースレベルのbaseLevel、カラー成分インデックスcldx、現在のピクチャの左上のサンプルを基準とした現在の変換ブロックの左上のサンプルを指定する輝度位置(x0、y0)、現在の係数スキャン位置(xC、yC)、変換ブロック幅の2進対数log2TbWidth、変換ブロックの高さの2進対数log2TbHeight、*sh_residual_coding_rice_shift*、および*sh_residual_coding_rice_offset*である。

このプロセスの出力は、ライスパラメータcRiceParamである。

成分インデックスcldxと左上の輝度位置(x0、y0)を持つ変換ブロックの配列AbsLevel[x][y]を指定すると、変数locSumAbsは次の疑似コードで指定されているように導出される。

```
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 ]
}
locSumAbs = Clip3( 0, 31, ( locSumAbs + sh_residual_coding_rice_offset ) >>
    sh_residual_coding_rice_shift ) - baseLevel * 5 )
```

変数locSumAbsを指定すると、テーブル4に示されるようにライスパラメータcRiceParamが導出される。

cRiceParam = cRiceParam + sh_residual_coding_rice_shift

baseLevelが0に等しい場合、変数ZeroPos[n]は次のように導出される。

ZeroPos[n] = (QState < 2 ? 1 : 2) << cRiceParam

【 0 3 3 1 】

【00311】別の例では、1つの制御フラグがシーケンスパラメータセット（またはシーケンスパラメータセット範囲の拡張シンタックス）で伝送され、変換スキップブロックのライスパラメータの伝送と、変換ブロック内のライスパラメータの導出のためのシフトパラメータおよび／またはオフセットパラメータの伝送が有効か無効かを示す。制御フラグが有効であると伝送されると、1つのシンタックス要素は、各変換スキップスライスに対してさらに伝送されて、そのスライスのライスパラメータを示し、2つのシンタックス要素は、各変換スライスに対してさらに伝送されて、そのスライスのライスパラメータを導出するためのシフトパラメータおよび／またはオフセットパラメータを示す。制御フラグが

無効として伝送されると（例えば、「0」に設定される）、変換スキップスライスのライスパラメータを示すために下位レベルでそれ以上のシンタックス要素は伝送されず、デフォルトのライスパラメータ（例えば、1）がすべての変換スキップスライスに使用され、変換スライスのライスパラメータ導出のためのシフトパラメータおよび／またはオフセットパラメータを示すために下位レベルでそれ以上のシンタックス要素は信号送信されず、デフォルトのシフトパラメータおよび／またはオフセットパラメータ（例えば、0）がすべての変換スライスに使用される。VVCドラフトに基づく対応する復号プロセスの例を以下に示す。ここで、THは、所定の値（例えば、0、1、2）である。VVCドラフトへの変更は、テーブル57に太字および斜体のフォントで示されている。*sh_residual_coding_rice_shift*、*sh_residual_coding_rice_offset*、および*sh_ts_residual_coding_rice_idx*は、異なる方法で符号化することができ、および／または最大値を有することができることは注目に値する。例えば、 $u(n)$ 、 n ビットを使用する符号なし整数、または $f(n)$ 、左ビットを最初に（左から右に）書き込まれた n ビットを使用する固定パターンビット文字列も、同じシンタックス要素の符号化／復号に使用できる。

【0332】

[00312]<シーケンスパラメータセットRBS Pシンタックス>

【0333】

【表57】

テーブル57. 残差コーディングのシンタックス

seq_parameter_set_rbsp() {	記述子
...	
sps_sign_data_hiding_enabled_flag	$u(1)$
<i>sps_residual_coding_info_present_in_sh_flag</i>	<i>$u(1)$</i>
sps_virtual_boundaries_enabled_flag	$u(1)$
...	
}	

【0334】

[00313]1に等しい*sps_residual_coding_info_present_in_sh_flag*は、SPSを参照するSHシンタックス構造内に*sh_residual_coding_rice_shift*、*sh_residual_coding_rice_offset*、*sh_ts_residual_coding_rice_idx*が存在し得ることを指定し、0に等しい*sps_residual_coding_info_present_in_sh_flag*は、SPSを参照するSHシンタックス構造内に*sh_residual_coding_rice_shift*、*sh_residual_coding_rice_offset*、*sh_ts_residual_coding_rice_idx*が存在しないことを指定する。*sps_residual_coding_info_present_in_sh_flag*が存在しない場合、*sps_residual_coding_info_present_in_sh_flag*の値は0に等しいと推定される。

【0335】

[00314]<スライスヘッダのシンタックス>

【0336】

【表58】

テーブル58. 残差コーディングのシンタックス

slice_header() {	記述子
...	
if(sps_transform_skip_enabled_flag && !sh_dep_quant_used_flag && !sh_sign_data_hiding_used_flag)	
sh_ts_residual_coding_disabled_flag	u(1)
if(sps_ts_residual_coding_rice_enabled_flag) {	
sh_residual_coding_rice_shift	ue(v)
sh_residual_coding_rice_offset	ue(v)
if(!sh_ts_residual_coding_disabled_flag)	
sh_ts_residual_coding_rice_index	ue(v)
}	
...	
}	

【0337】

[00315] sh_residual_coding_rice_shift は、abs_readinder[] および dec_abs_level[] のライスパラメータ導出プロセスに使用されるシフトパラメータを指定する。sh_residual_coding_rice_shift が存在しない場合、sh_residual_coding_rice_shift の値は0に等しいと推定される。

【0338】

[00316] sh_residual_coding_rice_offset は、abs_remainder[] および dec_abs_level[] のライスパラメータ導出プロセスに使用されるオフセットパラメータを指定する。sh_residual_coding_rice_offset が存在しない場合、sh_residual_coding_rice_offset の値は0に等しいと推定される。

【0339】

[00317] sh_ts_residual_coding_rice_idx は、residual_ts_coding() シンタックス構造に使用されるライスパラメータを指定する。sh_ts_residual_coding_rice_index が存在しない場合、sh_ts_residual_coding_rice_index の値は0に等しいと推定される。

【0340】

10

20

30

40

50

【表 5 9】

テーブル59. ライスパラメータ導出プロセス

abs_remainder[]の2値化プロセス	
...	
ライスパラメータcRiceParamは次のように導出される。	10
－ sps_ts_residual_coding_info_flagが1に等しく、transform_skip_flag[x0][y0][cldx]が1に等しく、sh_ts_residual_coding_disabled_flagが0に等しい場合、ライスパラメータcRiceParamは(sh_ts_residual_coding_rice_idx+TH)に等しく設定される。	
－ transform_skip_flag[x0][y0][cldx]が1に等しく、sh_ts_residual_coding_disabled_flagが0に等しい場合、ライスパラメータcRiceParamは1に等しく設定される。	
－ それ以外の場合、ライスパラメータcRiceParamは、変数baseLevelを4に設定して、カラー成分インデックスcldx、輝度位置(x0, y0)、現在の係数スキャン位置(xC, yC)、変換ブロック幅の2進対数log2Tbwidth、および変換ブロック高さの2進対数log2TbHeightを入力としてテーブル3に指定されているabs_remainder[]のライスパラメータ導出プロセスを呼び出すことによって導出される。	20
...	

【 0 3 4 1】

10

20

30

40

50

【表 6 0】

テーブル60. ライスパラメータ導出プロセス

abs_remainder[]とdec_abs_level[]のライスパラメータ導出プロセス

このプロセスへの入力は、ベースレベルのbaseLevel、カラー成分インデックスcldx、現在のピクチャの左上のサンプルを基準とした現在の変換ブロックの左上のサンプルを指定する輝度位置(x0, y0)、現在の係数スキャン位置(xC, yC)、変換ブロック幅の2進対数log2TbWidth、変換ブロックの高さの2進対数log2TbHeight、*sh_residual_coding_rice_shift*、および*sh_residual_coding_rice_offset*である。

このプロセスの出力は、ライスパラメータcRiceParamである。

成分インデックスcldxと左上の輝度位置(x0, y0)を持つ変換ブロックの配列AbsLevel[x][y]を指定すると、変数locSumAbsは次の疑似コードで指定されているように導出される。

```
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 ]
}
locSumAbs = Clip3( 0, 31, ( locSumAbs + sh_residual_coding_rice_offset ) >>
sh_residual_coding_rice_shift - baseLevel * 5 )
```

変数locSumAbsを指定すると、テーブル4に示されるようにライスパラメータcRiceParamが導出される。

cRiceParam = cRiceParam + sh_residual_coding_rice_shift

baseLevelが0に等しい場合、変数ZeroPos[n]は次のように導出される。

ZeroPos[n] = (QState < 2 ? 1 : 2) << cRiceParam

【 0 3 4 2】

[00318]さらに別の例では、1つのシンタックス要素は、各変換スキップスライスに対して伝送されて、そのスライスのライスパラメータを示し、2つのシンタックス要素は、各変換スライスに対して伝送されて、そのスライスのライスパラメータを導出するためのシフトパラメータおよび／またはオフセットパラメータを示す。VVCドラフトに基づく対応する復号プロセスの例を以下に示す。VVCドラフトへの変更は、テーブル61に太字と斜体のフォントで示されている。*sh_residual_coding_rice_shift*、*sh_residual_coding_rice_offset*、および*sh_residual_coding_rice_idx*は、異なる方法で符

号化することができ、ならびに／または最大値を有することができることは注目に値する。例えば、 $u(n)$ 、 n ビットを使用する符号なし整数、または $f(n)$ 、左ビットを最初に（左から右に）書き込まれた n ビットを使用する固定パターンビット文字列も、同じシンタックス要素の符号化／復号に使用できる。

【0343】

[00319]＜スライスヘッダのシンタックス＞

【0344】

【表61】

テーブル61. 残差コーディングのシンタックス

slice_header() {	記述子
...	
if(sps_transform_skip_enabled_flag && !sh_dep_quant_used_flag && !sh_sign_data_hiding_used_flag)	
sh_ts_residual_coding_disabled_flag	u(1)
<i>if(!sh_ts_residual_coding_disabled_flag)</i>	
<i>sh_ts_residual_coding_rice_idx</i>	<i>ue(v)</i>
<i>sh_residual_coding_rice_shift</i>	<i>ue(v)</i>
<i>sh_residual_coding_rice_offset</i>	<i>ue(v)</i>
...	
}	

【0345】

[00320] *sh_ts_residual_coding_rice_idx*は、*residual_ts_coding()* シンタックス構造に使用されるライスパラメータを指定する。*sh_ts_residual_coding_rice_idx*が存在しない場合、*sh_ts_residual_coding_rice_idx*の値は0に等しいと推定される。

【0346】

[00321] *sh_residual_coding_rice_offset*は、*abs_remainder[]* および *dec_abs_level[]* のライスパラメータ導出プロセスに使用されるオフセットパラメータを指定する。*sh_residual_coding_rice_offset*が存在しない場合、*sh_residual_coding_rice_offset*の値は0に等しいと推定される。

【0347】

[00322] *sh_ts_residual_coding_rice_idx*は、*residual_ts_coding()* シンタックス構造に使用されるライスパラメータを指定する。*sh_ts_residual_coding_rice_index*が存在しない場合、*sh_ts_residual_coding_rice_index*の値は0に等しいと推定される。

【0348】

10

20

30

40

50

【表 6 2】

テーブル62. ライスパラメータ導出プロセス

abs__remainder[]の2値化プロセス	
...	
ライスパラメータcRiceParamは次のように導出される。	
— transform__skip__flag[x0][y0][cldx]が1に等しく、sh__ts__residual__coding__disabled__flagが0に等しい場合、ライスパラメータcRiceParamはsh__ts__residual__coding__rice__idx+1に等しく設定される。	10
— それ以外の場合、ライスパラメータcRiceParamは、変数baseLevelを4に等しく設定して、カラー成分インデックスcldx、輝度位置(x0, y0)、現在の係数スキャン位置(xC, yC)、変換ブロック幅の2進対数log2Tbwidth、および変換ブロック高さの2進対数log2TbHeightを入力としてテーブル3に指定されているabs__remainder[]のライスパラメータ導出プロセスを呼び出すことによって導出される。	
...	20

【 0 3 4 9 】

30

40

50

【表 6 3】

テーブル63. ライスパラメータ導出プロセス

abs_remainder[]とdec_abs_level[]のライスパラメータ導出プロセス

このプロセスへの入力は、ベースレベルのbaseLevel、カラー成分インデックスcldx、現在のピクチャの左上のサンプルを基準とした現在の変換ブロックの左上のサンプルを指定する輝度位置(x0、y0)、現在の係数スキャン位置(xC、yC)、変換ブロック幅の2進対数log2TbWidth、変換ブロックの高さの2進対数log2TbHeight、*sh_residual_coding_rice_shift*、および*sh_residual_coding_rice_offset*である。

このプロセスの出力は、ライスパラメータcRiceParamである。

成分インデックスcldxと左上の輝度位置(x0、y0)を持つ変換ブロックの配列AbsLevel[x][y]を指定すると、変数locSumAbsは次の疑似コードで指定されているように導出される。

```
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 ]
}
locSumAbs = Clip3( 0, 31, ( locSumAbs + sh_residual_coding_rice_offset ) >>
    sh_residual_coding_rice_shift - baseLevel * 5 )
```

変数locSumAbsを指定すると、テーブル4に示されるようにライスパラメータcRiceParamが導出される。

cRiceParam = cRiceParam + sh_residual_coding_rice_shift

baseLevelが0に等しい場合、変数ZeroPos[n]は次のように導出される。

ZeroPos[n] = (QState < 2 ? 1 : 2) << cRiceParam

【 0 3 5 0】

【00323】さらに別の例では、1つの制御フラグは、ピクチャパラメータセット範囲の拡張シンタックスで伝送され、変換スキップブロックのライスパラメータの伝送と、変換ブロック内のライスパラメータの導出のためのシフトパラメータおよび／またはオフセットパラメータの伝送が有効であるか無効であるかを示す。制御フラグが有効であると伝送されると、1つのシンタックス要素は、そのピクチャの変換スキップ残差コーディングのためのライスパラメータを示すためにさらに伝送され、2つのシンタックス要素はさらに、そのピクチャのライスパラメータを導出するためのシフトパラメータおよび／またはオフセットパラメータを示すために正規の残差コーディングのために伝送される。制御フラグ

が無効として伝送されると（例えば、「0」に設定される）、変換スキップ残差コーディングのライスパラメータを示すために下位レベルでそれ以上のシンタックス要素は伝送されず、デフォルトのライスパラメータ（例えば、1）がすべての変換スキップ残差コーディングに使用され、正規の残差コーディングのライスパラメータを導出するためのシフトパラメータおよび／またはオフセットパラメータを示すために、下位レベルでそれ以上のシンタックス要素が伝送されることはなく、デフォルトのシフトパラメータおよび／またはオフセットパラメータ（例えば0）がすべての正規の残差コーディングに使用される。VVCドラフトに基づく対応する復号プロセスの例を以下に示す。ここで、THは、所定の値（例えば、0、1、2）である。VVCドラフトへの変更は、テーブル64に太字および斜体のフォントで示されている。pps_residual_coding_rice_shift、pps_residual_coding_rice_offset、およびpps_ts_residual_coding_rice_idxは、異なる方法で符号化することができ、および／または最大値を有することができることは注目に値する。例えば、u(n)、nビットを使用する符号なし整数、またはf(n)、nビットを使用し、左ビットから順に（左から右に）書き込まれた固定パターンのビット文字列は、同じシンタックス要素を符号化／復号するために使用することもできる。

【0351】

[00324]<ピクチャパラメータセット範囲の拡張シンタックス>

【0352】

【表64】

テーブル64. 残差コーディングのシンタックス

pps_range_extensions() {	記述子
...	
pps_residual_coding_info_flag	u(1)
if(pps_ts_residual_coding_rice_flag)	
pps_residual_coding_rice_shift	ue(v)
pps_residual_coding_rice_offset	ue(v)
pps_ts_residual_coding_rice_idx	ue(v)
}	
...	

【0353】

[00325]1に等しいpps_residual_coding_info_flagは、pps_residual_coding_rice_shift、pps_residual_coding_rice_offset、pps_ts_residual_coding_rice_indexが現在のピクチャに存在し得ることを指定し、0に等しいpps_residual_coding_info_flagは、pps_residual_coding_rice_shift、pps_residual_coding_rice_offset、およびpps_ts_residual_coding_rice_idxが現在のピクチャに存在しないことを指定する。pps_residual_coding_info_flagが存在しない場合、pps_residual_coding_info_flagの値は0に等しいと推定される。

【0354】

【00326】`pps_residual_coding_rice_shift`は、`abs_remainder[]` および `dec_abs_level[]` のライスパラメータ導出プロセスに使用されるシフトパラメータを指定する。`pps_residual_coding_rice_shift`が存在しない場合、`pps_residual_coding_rice_shift`の値は0に等しいと推定される。

【0355】

【00327】`pps_residual_coding_rice_offset`は、`abs_remainder[]` および `dec_abs_level[]` のライスパラメータ導出プロセスに使用されるオフセットパラメータを指定する。`pps_residual_coding_rice_offset`が存在しない場合、`pps_residual_coding_rice_offset`の値は0に等しいと推定される。

10

【0356】

【00328】`pps_ts_residual_coding_rice_idx`は、`residual_ts_coding()` シンタックス構造に使用されるライスパラメータを指定する。`pps_ts_residual_coding_rice_index`が存在しない場合、`pps_ts_residual_coding_rice_index`の値は0に等しいと推定される。

【0357】

【表65】

20

テーブル65. ライスパラメータ導出プロセス

`abs_remainder[]`の2値化プロセス

...

ライスパラメータ`cRiceParam`は次のように導出される。

- `pps_ts_residual_coding_rice_flag`が1に等しく、`transform_skip_flag[x0][y0][cldx]`が1に等しく、`sh_ts_residual_coding_disabled_flag`が0に等しい場合、ライスパラメータ`cRiceParam`は(`pps_ts_residual_coding_rice_idx + TH`)に等しく設定される。
- `transform_skip_flag[x0][y0][cldx]`が1に等しく、`sh_ts_residual_coding_disabled_flag`が0に等しい場合、ライスパラメータ`cRiceParam`は1に等しく設定される。
- それ以外の場合、ライスパラメータ`cRiceParam`は、変数`baseLevel`を4に設定して、カラー成分インデックス`cldx`、輝度位置(`x0`, `y0`)、現在の係数スキャン位置(`xC`, `yC`)、変換ブロック幅の2進対数`log2TbWidth`、および変換ブロック高さの2進対数`log2TbHeight`を入力としてテーブル3に指定されている`abs_remainder[]`のライスパラメータ導出プロセスを呼び出すことによって導出される。

30

...

40

【0358】

50

【表 6 6】

テーブル66. ライスパラメータ導出プロセス

abs_remainder[]とdec_abs_level[]のライスパラメータ導出プロセス

このプロセスへの入力は、ベースレベルのbaseLevel、カラー成分インデックスcldx、現在のピクチャの左上のサンプルを基準とした現在の変換ブロックの左上のサンプルを指定する輝度位置(x0、y0)、現在の係数スキャン位置(xC、yC)、変換ブロック幅の2進対数log2TbWidth、変換ブロックの高さの2進対数log2TbHeight、pps_residual_coding_rice_shift、およびpps_residual_coding_rice_offsetである。

このプロセスの出力は、ライスパラメータcRiceParamである。

成分インデックスcldxと左上の輝度位置(x0、y0)を持つ変換ブロックの配列AbsLevel[x][y]を指定すると、変数locSumAbsは次の疑似コードで指定されているように導出される。

```
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 ]
}
locSumAbs = Clip3( 0, 31, ( locSumAbs + pps_residual_coding_rice_offset ) >>
pps_residual_coding_rice_shift ) - baseLevel * 5 )
```

変数locSumAbsを指定すると、テーブル4に示されるようにライスパラメータcRiceParamが導出される。

cRiceParam = cRiceParam + pps_residual_coding_rice_shift

baseLevelが0に等しい場合、変数ZeroPos[n]は次のように導出される。

ZeroPos[n] = (QState < 2 ? 1 : 2) << cRiceParam

【0359】

[00329]本開示の第29の態様によれば、正規の残差コーディングにおいて、特定のシンタックス要素、例えば変換スキップ残差コーディングのabs_remainder、abs_remainder/dec_abs_levelに使用されるライスパラメータを導出するためのシフトパラメータとオフセットパラメータを符号化するために異なるライスパラメータを使用することと、現在のブロックの特定の符号化情報、例えばTB/CBおよび/またはスライス/プロファイルに関連付けられた量子化パラメータもしくは符号化ビット深度に従って、ならびに/あるいはTB/CB/スライス/ピクチャ/シーケンスレベルに関連付けられた新しいフラグ、例えば、sps_residual_co

`ding_info_j>resent_in_sh_flag`に従ってどれを使用するかを決定することが提案されている。

【0360】

【00330】一例では、1つの制御フラグはスライスヘッダで伝送され、変換スキップブロックのライスパラメータの導出プロセスと、変換ブロックのライスパラメータのシフトパラメータおよび／またはオフセットパラメータの導出プロセスが有効か無効かを示す。制御フラグが有効として伝送されると、ライスパラメータは、現在のブロックの特定の符号化情報、例えば、量子化パラメータおよびビット深度に従って、異なる条件で異なる可能性がある。そして、正規の残差コーディングにおけるライスパラメータ導出のためのシフトパラメータおよび／またはオフセットパラメータは、現在のブロックの特定の符号化情報、例えば、量子化パラメータおよびビット深度に従って、異なる条件で異なり得る。制御フラグが無効として伝送されると（例えば、「0」に設定される）、デフォルトのライスパラメータ（例えば、1）がすべての変換スキップスライスに使用され、デフォルトのシフトパラメータおよび／またはオフセットパラメータ（例えば、0）がすべての変換スライスに使用される。VVCドラフトに基づく対応する復号プロセスの例を以下に示す。ここで、 TH_A および TH_B は、所定の閾値（例えば、 $TH_A=8$ 、 $TH_B=18$ または 19 ）である。VVCドラフトへの変更は、テーブル67に太字と斜体のフォントで示されている。実際には、同じロジックが異なる方法で実装され得ることは注目に値する。例えば、特定の方程式またはルックアップテーブルを使用して、同じライスパラメータを導出することもできる。

【0361】

【00331】<スライスヘッダシンタックス>

【0362】

【表67】

テーブル67. 残差コーディングのシンタックス

<code>slice_header() {</code>	記述子
<code>...</code>	
<code>if(sps_transform_skip_enabled_flag && !sh_dep_quant_used_flag && !sh_sign_data_hiding_used_flag)</code>	
<code>sh_ts_residual_coding_disabled_flag</code>	<code>u(1)</code>
<i>sh_residual_coding_rice_flag</i>	<i>u(1)</i>
<code>...</code>	

【0363】

【00332】1に等しい`sh_residual_coding_rice_flag`は、ビット深度に依存するライスパラメータ導出プロセスが現在のスライスにおいて使用されることを指定し、0に等しい`sh_residual_coding_rice_flag`は、ビット深度に依存するライスパラメータ導出プロセスが現在のスライスにおいて使用されないことを指定する。

【0364】

10

20

30

40

50

【表 6 8】

テーブル68. ライスパラメータ導出プロセス

abs__remainder[]の2値化プロセス	
...	
ライスパラメータcRiceParamは次のように導出される。	
— sh_residual_coding_rice_flagが1に等しく、transform_skip_flag[x0][y0][cldx]は1に等しく、sh_ts_residual_coding_disabled_flagは0に等しい場合、ライスパラメータcRiceParamは$\text{Clip3}(1, \text{BitDepth} - TH_A, (TH_B - QP) / 6)$に等しく設定される。	10
— transform_skip_flag[x0][y0][cldx]が1に等しく、sh_ts_residual_coding_disabled_flagが0に等しい場合、ライスパラメータcRiceParamは1に等しく設定される。	
— それ以外の場合、ライスパラメータcRiceParamは、変数baseLevelを4に設定して、カラー成分インデックスcldx、輝度位置(x0, y0)、現在の係数スキャン位置(xC, yC)、変換ブロック幅の2進対数log2Tbwidth、および変換ブロック高さの2進対数log2TbHeightを入力としてテーブル3に指定されているabs__remainder[]のライスパラメータ導出プロセスを呼び出すことによって導出される。	20
...	

【 0 3 6 5】

30

40

50

【表 6 9 - 1】

テーブル69. ライスパラメータ導出プロセス

abs_remainder[]とdec_abs_level[]のライスパラメータ導出プロセス

このプロセスへの入力は、ベースレベルのbaseLevel、カラー成分インデックスcldx、現在のピクチャの左上のサンプルを基準とした現在の変換ブロックの左上のサンプルを指定する輝度位置(x0, y0)、現在の係数スキャン位置(xC, yC)、変換ブロック幅の2進対数log2TbWidth、変換ブロックの高さの2進対数log2TbHeight、*sh_residual_coding_rice_flag*である。

このプロセスの出力は、ライスパラメータcRiceParamである。

10

成分インデックスcldxと左上の輝度位置(x0, y0)を持つ変換ブロックの配列AbsLevel[x][y]を指定すると、変数locSumAbsは次の疑似コードで指定されているように導出される。

ShiftRice = *sh_residual_coding_rice_flag* ? (*BitDepth* > 10) ? Floor(Log2(4 * (*Bitdepth* - 10))) : 0 : 0

OffsetRice = *sh_residual_coding_rice_flag* ? (*ShiftRice* > 0) ? (1 << (*ShiftRice* - 1)) : 0 : 0

locSumAbs = 0

20

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]

(1494)

}

if(yC < (1 << log2TbHeight) - 1) {

30

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

if(yC < (1 << log2TbHeight) - 2)

locSumAbs += AbsLevel[xC][yC + 2]

}

locSumAbs = Clip3(0, 31, (locSumAbs + *OffsetRice*) >>

ShiftRice) - baseLevel * 5)

40

50

【表 6 9 - 2】

変数locSumAbsを指定すると、テーブル4に示されるようにライスパラメータcRiceParamが導出される。

$$cRiceParam = cRiceParam + ShiftRice$$

BaseLevelが0に等しい場合、変数ZeroPos[n]は次のように導出される。

$$ZeroPos[n] = (QState < 2 ? 1 : 2) << cRiceParam$$

10

【0 3 6 6】

[00333]さらに別の例では、VVCドラフトに基づく対応する復号プロセスが以下のように示される。ここで、THは、所定の閾値（例えば、18、19）である。VVCドラフトへの変更は、テーブル70に太字および斜体のフォントで示されている。実際には、同じロジックが異なる方法で実装され得ることは注目に値する。例えば、特定の方程式またはルックアップテーブルを使用して、同じライスパラメータを導出することもできる。

【0 3 6 7】

20

30

40

50

【表 7 0 - 1】

テーブル70. ライスパラメータ導出プロセス

abs_remainder[]の2値化プロセス ... ライスパラメータcRiceParamは次のように導出される。 — <i>BitDepth</i>が10より大きく、<i>transform_skip_flag[x0][y0][cldx]</i>が1に等しく、<i>sh_ts_residual_coding_disabled_flag</i>が0に等しい場合、ライスパラメータcRiceParamの導出プロセスは次のように指定される。 ライスパラメータ = $\text{Clip3}(1, 8, (\text{TH} - \text{QP}) / 6)$ — <i>transform_skip_flag[x0][y0][cldx]</i>が1に等しく、<i>sh_ts_residual_coding_disabled_flag</i>が0に等しい場合、ライスパラメータcRiceParamは1に等しく設定される。 — それ以外の場合、ライスパラメータcRiceParamは、変数baseLevelを4に設定して、カラー成分インデックスcldx、輝度位置(x0, y0)、現在の係数スキャン位置(xC, yC)、変換ブロック幅の2進対数log2Tbwidth、および変換ブロック高さの2進対数log2TbHeightを入力としてテーブル3に指定されているabs_remainder[]のライスパラメータ導出プロセスを呼び出すことによって導出される。	10
abs_remainder[]とdec_abs_level[]のライスパラメータ導出プロセス このプロセスへの入力は、ベースレベルのbaseLevel、カラー成分インデックスcldx、現在のピクチャの左上のサンプルに対する現在の変換ブロックの左上のサンプルを指定する輝度位置(x0, y0)、現在の係数スキャン位置(xC, yC)、変換ブロックの幅の2進対数log2TbWidth、変換ブロックの高さの2進対数log2TbHeightである。 このプロセスの出力は、ライスパラメータcRiceParamである。 成分インデックスcldxと左上の輝度位置(x0, y0)を持つ変換ブロックの配列AbsLevel[x][y]を指定すると、変数locSumAbsは次の疑似コードで指定されているように導出される。 $\text{ShiftRice} = (\text{BitDepth} > 10) ? \text{Floor}(\text{Log2}(4 * (\text{Bitdepth} - 10))) : 0$ $\text{OffsetRice} = (\text{BitDepth} > 10) ? (\text{ShiftRice} > 0) ? (1 << (\text{ShiftRice} - 1)) : 0 : 0$ locSumAbs = 0	30

40

50

【表 7 0 - 2】

```

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 ]
}
locSumAbs = Clip3( 0, 31, ( ( locSumAbs + OffsetRice ) >>
ShiftRice ) - baseLevel * 5 )

```

(1494)

10

20

変数locSumAbsを指定すると、テーブル4に示されるようにライスパラメータcRiceParamが導出される。

cRiceParam = cRiceParam + ShiftRice

baseLevelが0に等しい場合、変数ZeroPos[n]は次のように導出される。

ZeroPos[n] = (QState < 2 ? 1 : 2) << cRiceParam

30

【 0 3 6 8 】

[00334]本開示の別の態様によれば、これらの上記のコーディングツールの値が、一般的な制約情報における他のものと同じ一般的な制約制御を提供するフラグを立てるという制約を追加することが提案される。

【 0 3 6 9 】

[00335]例えば、1に等しいsps_ts_residual_coding_rice_present_in_sh_flagは、SPSを参照するSHシンタックス構造内にsh_ts_residual_coding_rice_idxが存在し得ることを指定する。0に等しいsps_ts_residual_coding_rice_present_in_sh_flagは、SPSを参照するSHシンタックス構造内にsh_ts_residual_coding_rice_idxが存在しないことを指定する。この開示によれば、他のフラグと同じ一般的な制約制御を提供するために、一般的な制約情報シンタックスにシンタックス要素gci_no_ts_residual_coding_rice_constraint_flagを追加することが提案されている。VVCドラフトの復号プロセスの例を以下に示す。VVCドラフトへの変更が強調表示される。追加された部分は斜体で強調表示される。

40

【 0 3 7 0 】

50

【表 7 1】

general_constraint_info() {	記述子
...	
<i>gci_no_ts_residual_coding_rice_constraint_flag</i>	<i>u(1)</i>
...	
}	

*gci_no_ts_residual_coding_rice_constraint_flag*が1に等しい場合は、
*sps_ts_residual_coding_rice_flag*が0に等しいものとして指定する。
*gci_no_ts_residual_coding_rice_constraint_flag*が0に等しい場合、そのような制約は課されない。

10

【0371】

[00336]図19は、本開示の一例によるビデオコーディングのための方法を示す。この方法は、例えば、デコーダに適用することができる。ステップ1902において、デコーダは、シーケンスパラメータセット（SPS）を参照するSHシンタックス構造内にインデックス *sh_ts_residual_coding_rice_idx* が存在するかどうかを示すSPS残差符号化フラグを受信することができる。

20

【0372】

[00337]ステップ1904において、SPS残差符号化フラグの値が1に等しいとの決定に回答して、デコーダは、SPSを参照するスライスヘッド（SH）シンタックス構造内に *sh_ts_residual_coding_rice_idx* が存在すると決定することができる。

【0373】

[00338]ステップ1906において、残差符号化フラグの値が0に等しいとの決定に回答して、デコーダは、SPSを参照するSHシンタックス構造内に *sh_ts_residual_coding_rice_idx* が存在しないと決定することができる。

30

【0374】

[00339]別の例では、*pps_ts_residual_coding_rice_flag*が1に等しい場合は、*pps_ts_residual_coding_rice_index*が現在のピクチャに存在し得ることを指定し、*pps_ts_residual_coding_rice_flag*が0に等しい場合は、*pps_ts_residual_coding_rice_idx*が現在のピクチャに存在しないことを指定する。この開示によれば、他のフラグと同じ一般的な制約制御を提供するために、一般的な制約情報シンタックスにシンタックス要素 *gci_no_ts_residual_coding_rice_constraint_flag* を追加することが提案されている。VVCドラフトの復号プロセスの例を以下に示す。VVCドラフトへの変更が強調表示される。追加された部分は斜体で強調表示される。

40

【0375】

50

【表 7 2】

general_constraint_info() {	記述子
...	
<i>gci_no_ts_residual_coding_rice_constraint_flag</i>	<i>u(1)</i>
...	
}	

10

*gci_no_ts_residual_coding_rice_constraint_flag*が1に等しい場合は、
*pps_ts_residual_coding_rice_flag*が0に等しいものとするを指定する。
*gci_no_ts_residual_coding_rice_constraint_flag*が0に等しい場合、そのような制約は課されない。

【0376】

[00340]さらに別の例では、1に等しい*sps__rice_adaptation_enabled__flag*は、*abs__remaining[]*および*dec__abs__level*の2値化のためのライスパラメータが式によって導出され得ることを示す。

20

【0377】

[00341]式には、 $RiceParam = RiceParam + shiftVal$ および
 $shiftVal = (localSumAbs < Tx[0]) ? Rx[0] : ((localSumAbs < Tx[1]) ? Rx[1] : ((localSumAbs < Tx[2]) ? Rx[2] : ((localSumAbs < Tx[3]) ? Rx[3] : Rx[4])))$ が含まれ得、リスト*Tx[]*および*Rx[]*は次のように指定される：*Tx[]* = {32, 128, 512, 2048} >> (1523) *Rx[]* = {0, 2, 4, 6, 8}

【0378】

[00342]図20は、本開示の一例によるビデオコーディングのための方法を示す。この方法は、例えば、デコーダに適用することができる。ステップ2002において、デコーダは、シンタックス*abs__remaining*および*dec__abs__level*の2値化のための代替ライスパラメータ導出が使用されるかどうかを示すシーケンスパラメータセット (SPS) 適応可能フラグを受信することができる。

30

【0379】

[00343]ステップ2004において、SPS適応可能フラグの値が1に等しいとの決定に応答して、デコーダは、シンタックスの2値化のための代替ライスパラメータ導出が使用されると決定することができる。

【0380】

[00344]ステップ2006において、SPS適応可能フラグの値が0に等しいとの決定に応答して、デコーダは、シンタックスの2値化のための代替ライスパラメータ導出が使用されないと決定することができる。

40

【0381】

[00345]本開示によれば、他のフラグと同じ一般的な制約制御を提供するために、一般的な制約情報シンタックスにシンタックス要素*gci__no__rice__adaptation__constraint__flag*を追加することが提案される。VVCドラフトの復号プロセスの例を以下に示す。VVCドラフトへの変更が強調表示される。追加された部分は斜体で強調表示される。

【0382】

50

【表 7 3】

general_constraint_info() {	記述子
...	
<i>gci_no_rice_adaptation_constraint_flag</i>	<i>u(1)</i>
...	
}	

*gci_no_rice_adaptation_constraint_flag*が1に等しい場合は、
*sps_rice_adaptation_enabled_flag*が0に等しいものとして指定する。
*gci_no_rice_adaptation_constraint_flag*が0に等しい場合、そのような制約は課されない。

10

【0383】

[00346]図21は、本開示の一例によるビデオコーディングのための方法を示す。この方法は、例えば、デコーダに適用することができる。ステップ2102において、デコーダは、他のフラグに対する一般的な制約制御を提供するために残差符号化ライス制約フラグを受信することができる。

20

【0384】

[00347]ステップ2104において、残差符号化ライス制約フラグの値が1に等しいとの決定に応答して、デコーダは、他のフラグの値が0に等しいと決定することができる。

【0385】

[00348]提案されたライスパラメータ適応方式は変換スキップ残差コーディング（TSRC）にのみ使用されるため、提案された方法は、TSRCが有効化されている場合にのみ有効となり得る。同様に、本開示の1つまたは複数の実施形態では、変換スキップモードが一般的な制約情報レベルから無効になっている場合、例えば、*gci_no_transform_skip_constraint_flag*の値が1に設定されている場合に、*gci_no_rice_adaptation_constraint_flag*の値が1であることを要求する1つのbtiストリーム制約を追加することが提案されている。

30

【0386】

[00349]上記の方法は、1つまたは複数の回路を含む装置を使用して実装することができる。これには、特定用途向け集積回路（ASIC）、デジタル信号プロセッサ（DSP）、デジタル信号処理デバイス（DSPD）、プログラマブルロジックデバイス（PLD）、フィールドプログラマブルゲート配列（FPGA）、コントローラ、マイクロコントローラ、マイクロプロセッサ、またはその他の電子構成要素が含まれる。この装置は、上述の方法を実行するために、他のハードウェアまたはソフトウェア構成要素と組み合わせて回路を使用することができる。上記で開示された各モジュール、サブモジュール、ユニット、またはサブユニットは、1つまたは複数の回路を使用して少なくとも部分的に実装され得る。

40

【0387】

[00350]本開示の他の例は、本明細書に開示される本開示の明細書および実施を考慮することにより、当業者には明らかとなるであろう。本出願は、その一般原理に従い、当技術分野で既知または慣例の範囲内にある本開示からのそのような逸脱を含む、本開示のあらゆる変形、使用、または適応を網羅することを意図している。明細書および実施例は、例示としてのみ考慮されることを意図している。

【0388】

50

本開示は、上で説明し、添付の図面に示した正確な例に限定されず、本開示の範囲から逸脱することなくさまざまな修正および変更を行うことができることが理解されるであろう。

【0389】

[00351]図18は、ユーザインターフェース1860と結合されたコンピューティング環境1810を示す。コンピューティング環境1810は、データ処理サーバーの一部とすることができる。コンピューティング環境1810は、プロセッサ1820、メモリ1840、およびI/Oインターフェース1850を含む。

【0390】

[00352]プロセッサ1820は、通常、表示、データ取得、データ通信、および画像処理に関連する操作など、コンピューティング環境1810の全体的な操作を制御する。プロセッサ1820は、上述の方法におけるステップのすべてまたは一部を実行するための命令を実行する1つまたは複数のプロセッサを含み得る。さらに、プロセッサ1820は、プロセッサ1820と他の構成要素との間の対話を容易にする1つまたは複数のモジュールを含み得る。プロセッサは、中央処理装置（CPU）、マイクロプロセッサ、シングルチップマシン、GPUなどであってもよい。

【0391】

[00353]メモリ1840は、コンピューティング環境1810の操作をサポートするために、さまざまなタイプのデータを記憶するように構成されている。メモリ1840は、所定のソフトウェア1842を含み得る。そのようなデータの例には、コンピューティング環境1810上で操作する任意のアプリケーションまたは方法のための命令、ビデオデータセット、画像データなどが含まれる。メモリ1840は、スタティックランダムアクセスメモリ（SRAM）、電氣的に消去可能プログラマブル読み取り専用メモリ（EEPROM）、消去可能プログラマブル読み取り専用メモリ（EPROM）、プログラマブル読み取り専用メモリ（PROM）、読み取り専用メモリ（ROM）、磁気メモリ、フラッシュメモリ、磁気ディスクまたは光ディスクなどの任意のタイプの揮発性もしくは不揮発性メモリデバイス、またはそれらの組み合わせを使用することによって実装され得る。

【0392】

[00354]I/Oインターフェース1850は、プロセッサ1820と、キーボード、クリックホイール、ボタンなどの周辺インターフェースモジュールとの間のインターフェースを提供する。ボタンには、ホームボタン、スキャン開始ボタン、スキャン停止ボタンが含まれるが、これらに限定されない。I/Oインターフェース1850は、エンコーダおよびデコーダと結合することができる。

【0393】

[00355]いくつかの実施形態では、上記の方法を実行するための、コンピューティング環境1810内のプロセッサ1820によって実行可能な、メモリ1840に含まれるような複数のプログラムを含む非一時的なコンピュータ可読記憶媒体も提供される。例えば、非一時的なコンピュータ可読記憶媒体は、ROM、RAM、CD-ROM、磁気テープ、フロッピーディスク、光データ記憶デバイスなどであってもよい。

【0394】

[00356]非一時的なコンピュータ可読記憶媒体は、1つまたは複数のプロセッサを有するコンピューティングデバイスによって実行される複数のプログラムを記憶しており、複数のプログラムは、1つまたは複数のプロセッサによって実行されると、コンピューティングデバイスに、上述の動き予測方法を実行させる。

【0395】

[00357]いくつかの実施形態では、コンピューティング環境1810は、上記の方法を実行するため1つまたは複数の特定用途向け集積回路（ASIC）、デジタル信号プロセッサ（DSP）、デジタル信号処理デバイス（DSPD）、プログラマブルロジックデバイス（PLD）、フィールドプログラマブルゲート配列（FPGA）、グラフィック処理ユニット（GPU）、コントローラ、マイクロコントローラ、マイクロプロセッサ、また

10

20

30

40

50

はその他の電子構成要素を用いて実装され得る。

【0396】

[00358]本開示の説明は、例示を目的として提示されたものであり、網羅的であること、または本開示に限定されることを意図したものではない。前述の説明および関連する図面に提示された教示の恩恵を受ける当業者には、多くの修正、変形、および代替の実装が明らかとなるであろう。

【0397】

[00359]実施例は、本開示の原理を説明し、他の当業者がさまざまな実装に関して本開示を理解できるようにするため、基礎となる原理とさまざまな実装を、企図される特定の用途に適したようにさまざまな変更を加えて最大限に利用するために選択され、説明されたものである。したがって、本開示の範囲は、開示された実装の特定の例に限定されるものではなく、修正および他の実装も本開示の範囲内に含まれることが意図されることが理解されるべきである。

10

20

30

40

50

【図 3 C】

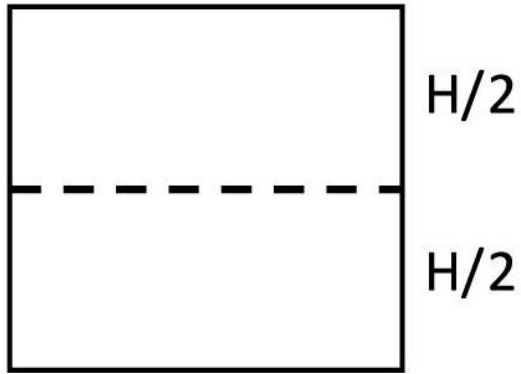


図3C

【図 3 D】

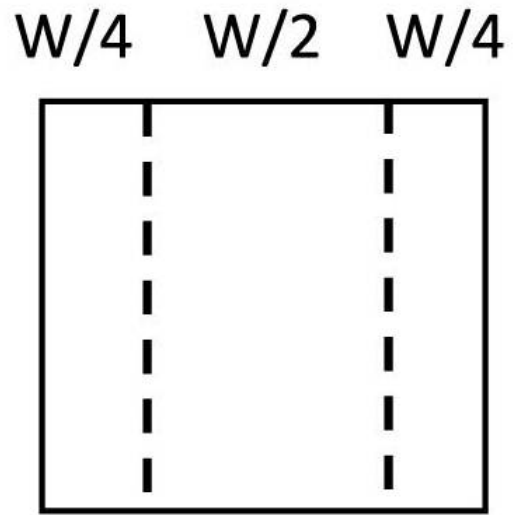


図3D

【図 3 E】

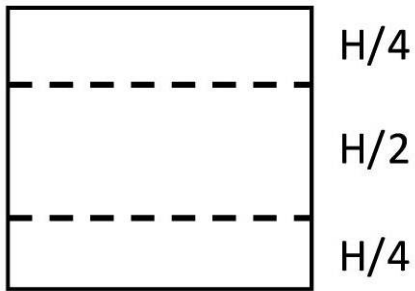


図3E

【図 4】

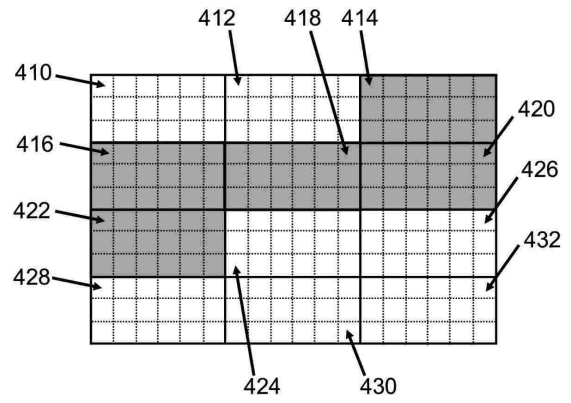


図4

10

20

30

40

50

【図 5】

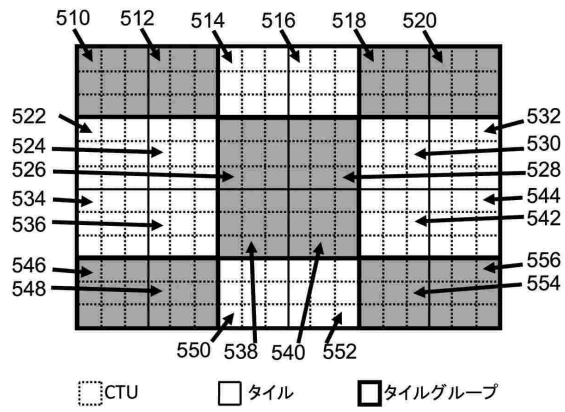
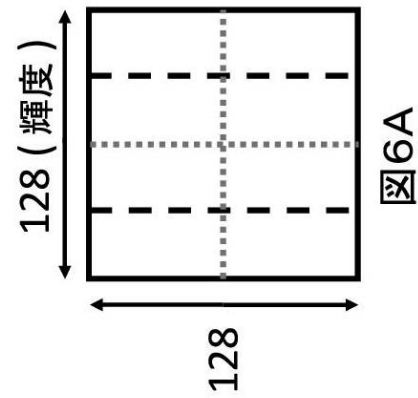


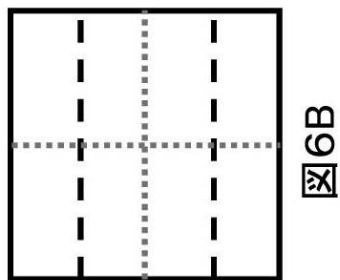
図5

【図 6 A】

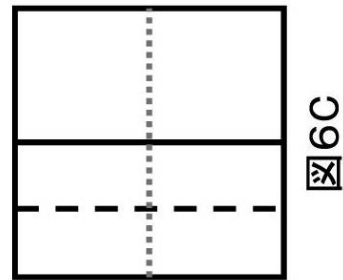


10

【図 6 B】



【図 6 C】



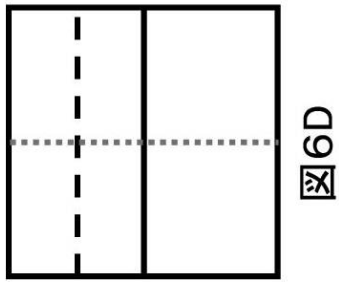
20

30

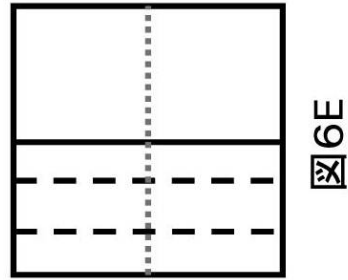
40

50

【図 6 D】

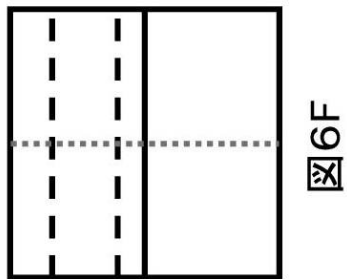


【図 6 E】

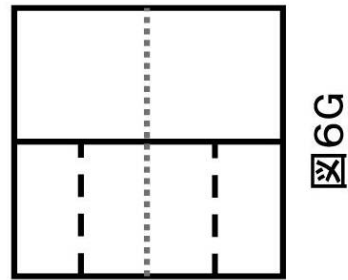


10

【図 6 F】



【図 6 G】



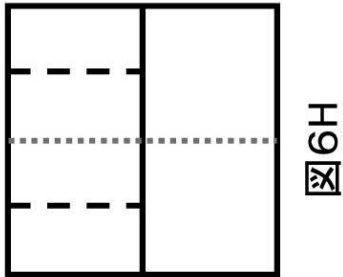
20

30

40

50

【図 6 H】



【図 7】

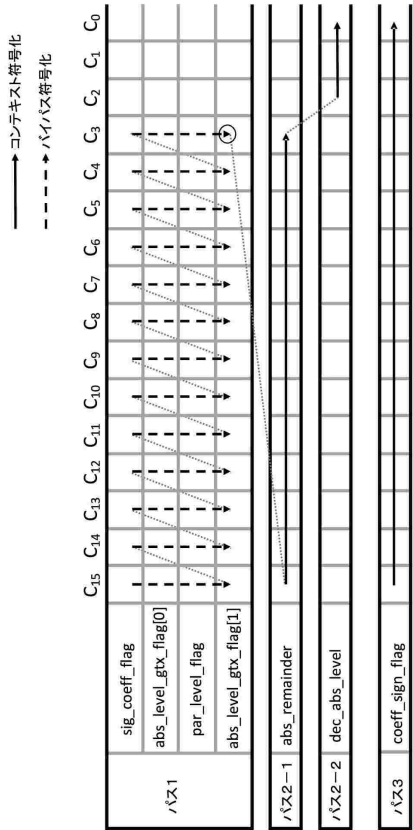


図 7

【図 8】

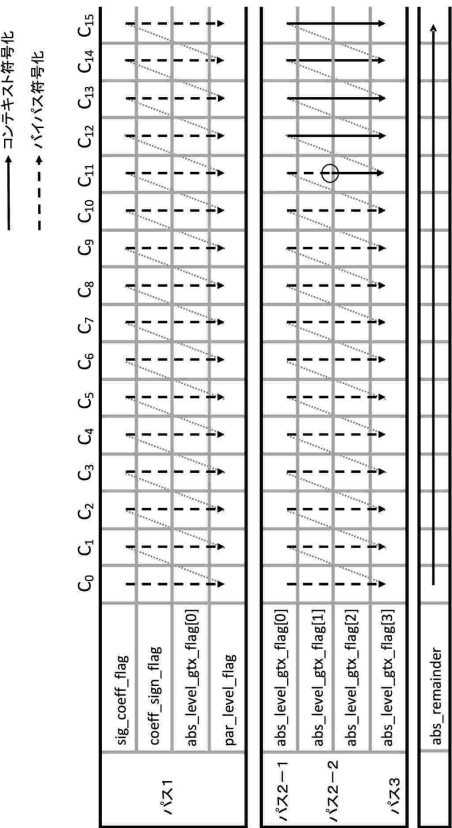


図 8

【図 9】

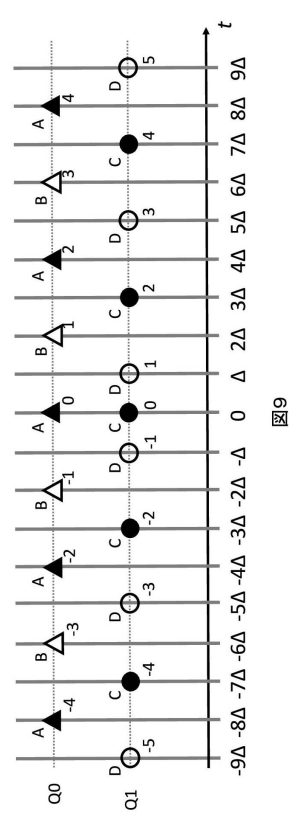


図 9

【図 10 A】

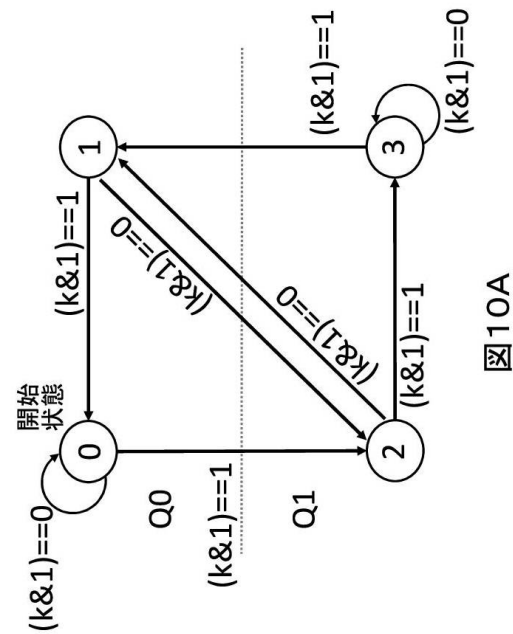


図 10A

【図 10 B】

現在の状態	...への次の状態			
	$(k \& 1) == 0$	$(k \& 1) == 1$	$(k \& 1) == 0$	$(k \& 1) == 1$
0	0	2	2	0
1	2	0	0	3
2	1	3	3	1
3	3	1	1	2

図 10B

【図 1 1】

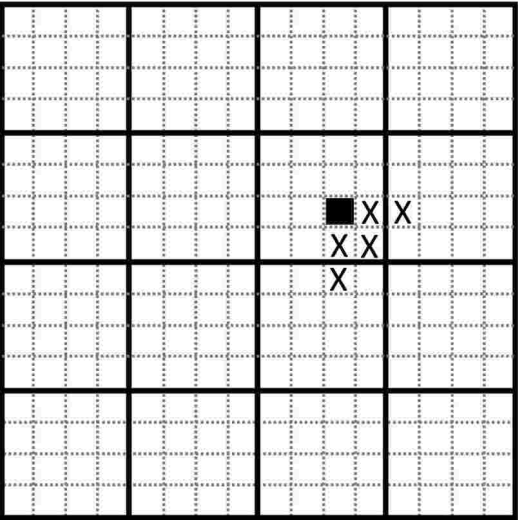


図 11

【図 1 2】

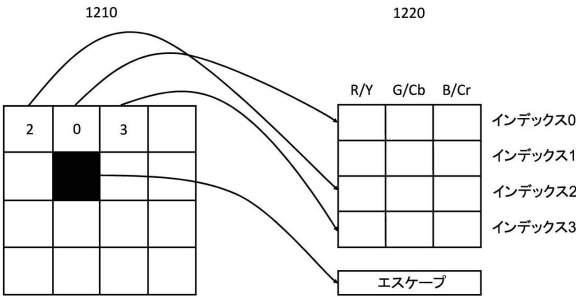


図 12

10

20

30

40

50

【図 1 3】

1310

インデックス	G/Y	B/Cb	R/Cr
0	G0	B0	R0
1	G1	B1	R1
2	G2	B2	R2
3	G3	B3	R3
4	G4	B4	R4
5	G5	B5	R5

1320

予測フラグ	インデックス	G/Y	B/Cb	R/Cr
1	0	G0	B0	R0
0				
1	1	G2	B2	R2
1	2	G3	B3	R3
0				
0				

再使用されたパレットエントリ(3)

新しいパレットエントリ(2), 信号伝達される

	3	G3N	B3N	R3N
	4	G4N	B4N	R4N

図 13

【図 1 4 A】

0	0	0	0	0	0	0	0
0	1	1	1	1	1	0	0
0	1	1	1	1	1	0	0
0	0		1	1	0	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

図14A

【図 1 4 B】

0	0	0	0	0	0	0	0
0	1	1	1	1	1	0	0
0	1	1	1	1	1	0	0
0	0		1	1	0	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

図14B

【図 1 5 A】

m							

図15A

【図 15 B】

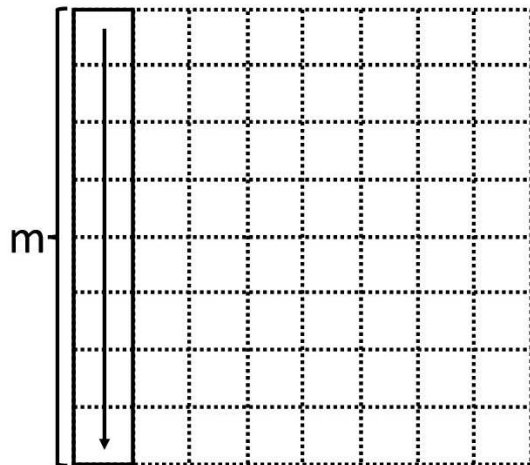


図 15B

【図 16】

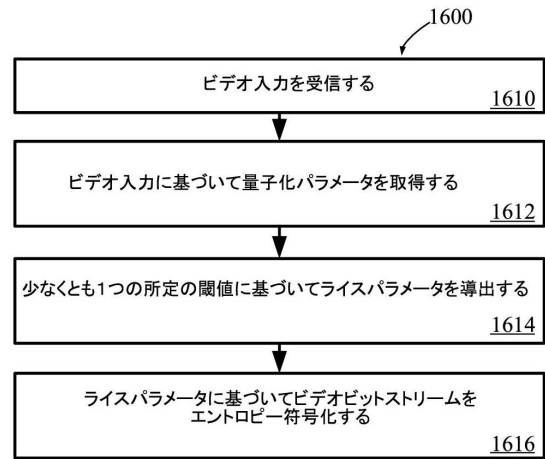


図 16

【図 17】

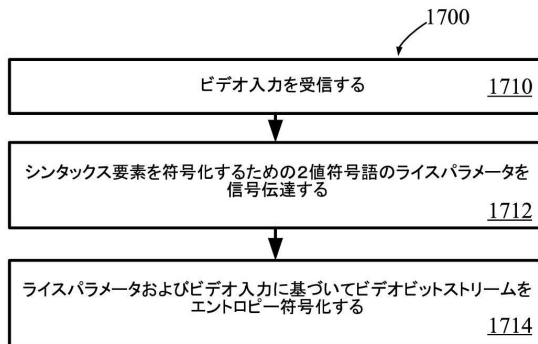


図 17

【図 18】

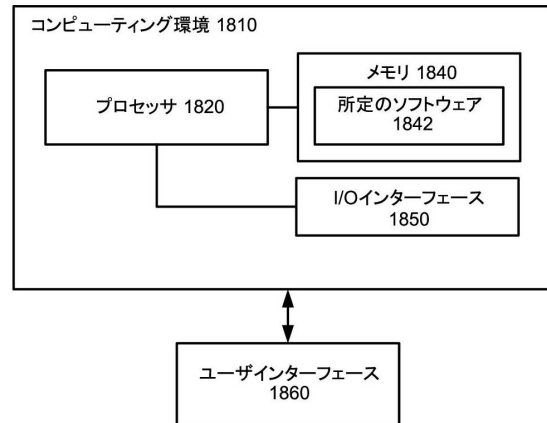


図 18

10

20

30

40

50

【図 19】

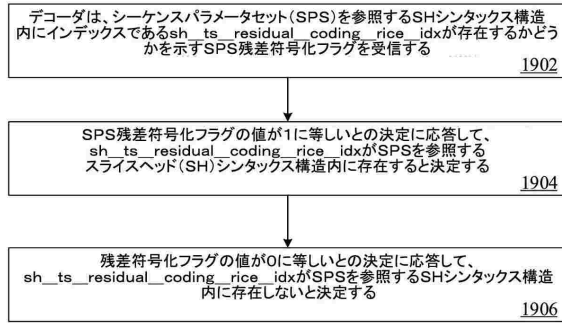


図 19

【図 20】

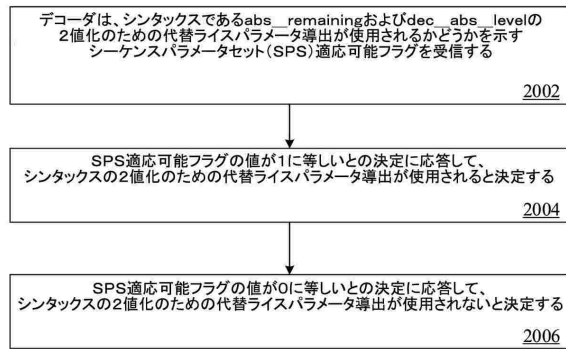


図 20

【図 21】

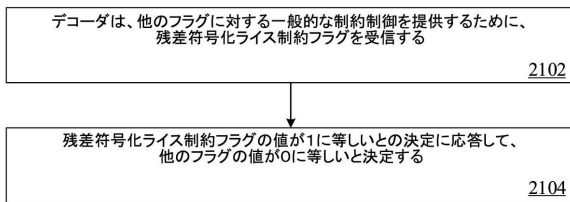


図 21

10

20

30

40

50

フロントページの続き

(74)代理人 100138759
弁理士 大房 直樹

(74)代理人 100201743
弁理士 井上 和真

(72)発明者 ジュ, ホーンーージェン
アメリカ合衆国 9 2 1 2 2, サンディエゴ, ユニバーシティ・センター・レーン 8 9 1 0, スイート 4 0 0

(72)発明者 シウ, シャオユー
アメリカ合衆国カリフォルニア州 9 2 1 3 0, サンディエゴ, アリソ・バレー・ウェイ 6 6 9 1

(72)発明者 チェン, イーウエン
アメリカ合衆国カリフォルニア州 9 2 1 2 2, サンディエゴ, ユニバーシティ・センター・レーン 8 9 1 0, スイート 4 0 0

(72)発明者 チェン, ウエイ
アメリカ合衆国カリフォルニア州 9 2 1 3 0, サンディエゴ, カメル・キャニオン・ロード 1 2 5 6 9

(72)発明者 クオ, チェーウェイ
中華人民共和国 ベイジン, ハイディアン・ディストリクト, シャンディ・ウエスト・ロード, ナンバー 6, ビルディング 1, ファースト・フロア, ルーム 1 0 1 ディー1-7

(72)発明者 ワン, シアーンリン
アメリカ合衆国カリフォルニア州 9 2 1 2 2, サンディエゴ, ユニバーシティ・センター・レーン 8 9 1 0, スイート 4 0 0

(72)発明者 ユ, ビーン
中華人民共和国 ベイジン, ハイディアン・ディストリクト, シャンディ・ウエスト・ロード, ナンバー 6, ビルディング 1, ファースト・フロア, ルーム 1 0 1 ディー1-7

審査官 松元 伸次

(56)参考文献 Hong-Jheng Jhu ET AL.; CE-2.1: Slice based Rice parameter selection for transform skip residual coding, Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29 21st Meeting, by teleconference, 6-15 Jan. 2021, JVET-U0075-v2, Joint Video Experts Team (JVET), 2020年12月31日, pp.4-5

(58)調査した分野 (Int.Cl., DB名)
H 0 4 N 1 9 / 0 0 - 1 9 / 9 8