(19)日本国特許庁(JP)

# (12)特許公報(B2)

(11)特許番号
特許第7682108号
(P7682108)

(45)発行日　令和7年5月23日(2025.5.23)

(24)登録日　令和7年5月15日(2025.5.15)

| (51)国際特許分類 | | FI | |
|---|---|---|---|
| *H04N* | *19/129 (2014.01)* | H04N | 19/129 |
| *H04N* | *19/13 (2014.01)* | H04N | 19/13 |
| *H04N* | *19/18 (2014.01)* | H04N | 19/18 |
| *H04N* | *19/64 (2014.01)* | H04N | 19/64 |
| *H04N* | *19/70 (2014.01)* | H04N | 19/70 |

請求項の数　6　（全43頁）

(21)出願番号　特願2021-571049(P2021-571049)
(86)(22)出願日　令和2年5月12日(2020.5.12)
(65)公表番号　特表2022-538968(P2022-538968A)
(43)公表日　令和4年9月7日(2022.9.7)
(86)国際出願番号　PCT/US2020/032460
(87)国際公開番号　WO2020/263442
(87)国際公開日　令和2年12月30日(2020.12.30)
審査請求日　令和5年4月28日(2023.4.28)
(31)優先権主張番号　62/865,916
(32)優先日　令和1年6月24日(2019.6.24)
(33)優先権主張国・地域又は機関
米国(US)
(31)優先権主張番号　62/953,460
(32)優先日　令和1年12月24日(2019.12.24)
最終頁に続く

(73)特許権者　511050697
アリババ グループ ホウルディング リミテッド
英国領ケイマン諸島 グランド ケイマン ジョージ タウン ピーオーボックス 847 ワン キャピタル プレイス フォース フロア
(74)代理人　100079108
弁理士　稲葉 良幸
(74)代理人　100109346
弁理士　大貫 敏史
(74)代理人　100117189
弁理士　江口 昭彦
(74)代理人　100134120
弁理士　内藤 和彦
最終頁に続く

(54)【発明の名称】　ビデオデータの変換スキップ残差符号化

(57)【特許請求の範囲】

【請求項１】

ビデオデータのデコーダによって実施される復号方法であって、

ビデオフレームのサブブロックの複数の変換係数をスキャンする第１のパスを実行することを含み、

前記スキャンする第１のパスが、

コンテキスト符号化されたビンの残りの数が４以上であるか否かを決定することと、

前記コンテキスト符号化されたビンの残りの数が４以上であることに応じて、

対象の変換係数のｓｉｇ＿ｃｏｅｆｆ＿ｆｌａｇを復号することであって、前記ｓｉｇ＿ｃｏｅｆｆ＿ｆｌａｇは、対象の前記変換係数のレベルがゼロであるか否かを示すこと、及び

対象の前記変換係数の前記レベルがゼロではないことに応じて、対象の前記変換係数のｃｏｅｆｆ＿ｓｉｇｎ＿ｆｌａｇ及びａｂｓ＿ｌｅｖｅｌ＿ｇｔｘ＿ｆｌａｇ［０］を復号することであって、前記ｃｏｅｆｆ＿ｓｉｇｎ＿ｆｌａｇは、対象の前記変換係数の前記レベルの符号を表し、前記ａｂｓ＿ｌｅｖｅｌ＿ｇｔｘ＿ｆｌａｇ［０］は、対象の前記変換係数の前記レベルの絶対値が１より大であることを表すこと

を実行することと、を含む、復号方法。

【請求項２】

前記コンテキスト符号化されたビンの残りの数が４未満であることに応じて、前記スキャンする第１のパスを終了することを更に含む、請求項１に記載の復号方法。

【請求項３】
ビデオデータのエンコーダによって実施される符号化方法であって、
ビデオフレームのサブブロックの複数の変換係数をスキャンする第１のパスを実行することを含み、
前記スキャンする第１のパスが、
コンテキスト符号化されたビンの残りの数が４以上であるか否かを決定することと、
前記コンテキスト符号化されたビンの残りの数が４以上であることに応じて、
対象の変換係数のｓｉｇ＿ｃｏｅｆｆ＿ｆｌａｇを符号化することであって、前記ｓｉｇ＿ｃｏｅｆｆ＿ｆｌａｇは、対象の前記変換係数のレベルがゼロであるか否かを示すこと、及び
対象の前記変換係数の前記レベルがゼロではないことに応じて、対象の前記変換係数のｃｏｅｆｆ＿ｓｉｇｎ＿ｆｌａｇ及びａｂｓ＿ｌｅｖｅｌ＿ｇｔｘ＿ｆｌａｇ［０］を符号化することであって、前記ｃｏｅｆｆ＿ｓｉｇｎ＿ｆｌａｇは、対象の前記変換係数の前記レベルの符号を表し、前記ａｂｓ＿ｌｅｖｅｌ＿ｇｔｘ＿ｆｌａｇ［０］は、対象の前記変換係数の前記レベルの絶対値が１より大であることを表すこと
を実行することと、を含む、符号化方法。
【請求項４】
前記コンテキスト符号化されたビンの残りの数が４未満であることに応じて、前記スキャンする第１のパスを終了することを更に含む、請求項３に記載の符号化方法。
【請求項５】
ビデオデータを符号化するシステムであって、前記システムが、
一組の命令を格納する少なくとも１つのメモリと、
少なくとも１つのプロセッサと、を備え、
前記プロセッサは、前記一組の命令を実行して、ビデオフレームのサブブロックの複数の変換係数をスキャンする第１のパスを実行することを含む動作を前記システムに行わせるように構成され、前記スキャンする第１のパスが、
コンテキスト符号化されたビンの数が４以上であるか否かを決定することと、
前記コンテキスト符号化されたビンの残りの数が４以上であることに応じて、
対象の変換係数のｓｉｇ＿ｃｏｅｆｆ＿ｆｌａｇを符号化することであって、前記ｓｉｇ＿ｃｏｅｆｆ＿ｆｌａｇは、対象の前記変換係数のレベルがゼロであるか否かを示すこと、及び
対象の前記変換係数の前記レベルがゼロではないことに応じて、対象の前記変換係数のｃｏｅｆｆ＿ｓｉｇｎ＿ｆｌａｇ及びａｂｓ＿ｌｅｖｅｌ＿ｇｔｘ＿ｆｌａｇ［０］を符号化することであって、前記ｃｏｅｆｆ＿ｓｉｇｎ＿ｆｌａｇは、対象の前記変換係数の前記レベルの符号を表し、前記ａｂｓ＿ｌｅｖｅｌ＿ｇｔｘ＿ｆｌａｇ［０］は、対象の前記変換係数の前記レベルの絶対値が１より大であることを表すこと
を実行することと、を含む、システム。
【請求項６】
ビデオデータを復号するシステムであって、前記システムが、
一組の命令を格納する少なくとも１つのメモリと、
１つ又は複数のプロセッサと、を備え、
前記プロセッサは、前記一組の命令を実行して、ビデオフレームのサブブロックの複数の変換係数をスキャンする第１のパスを実行することを含む動作を前記システムに行わせるように構成され、前記スキャンする第１のパスが、
コンテキスト符号化されたビンの残りの数が４以上であるか否かを決定することと、
前記コンテキスト符号化されたビンの残りの数が４以上であることに応じて、
対象の変換係数のｓｉｇ＿ｃｏｅｆｆ＿ｆｌａｇを復号することであって、前記ｓｉｇ＿ｃｏｅｆｆ＿ｆｌａｇは、対象の前記変換係数のレベルがゼロであるか否かを示すこと、及び
対象の前記変換係数の前記レベルがゼロではないことに応じて、対象の前記変換係数

のｃｏｅｆｆ＿ｓｉｇｎ＿ｆｌａｇ及びａｂｓ＿ｌｅｖｅｌ＿ｇｔｘ＿ｆｌａｇ［０］を復号することであって、前記ｃｏｅｆｆ＿ｓｉｇｎ＿ｆｌａｇは、対象の前記変換係数の前記レベルの符号を表し、前記ａｂｓ＿ｌｅｖｅｌ＿ｇｔｘ＿ｆｌａｇ［０］は、対象の前記変換係数の前記レベルの絶対値が１より大であることを表すこと
を実行することと、を含む、システム。

【発明の詳細な説明】

【技術分野】

【０００１】

関連出願の相互参照

［0001］本開示は、２０１９年６月２４日に出願された米国仮特許出願第６２／８６５，９１６号、２０１９年９月１８日に出願された同第６２／９０２，１１５号、及び２０１９年１２月２４日に出願された同第６２／９５３，４６０号の優先権及び優先権の利益を主張するものである。上記３つの仮出願は、その全体が参照により本明細書に組み込まれる。

【０００２】

技術分野

［0002］本開示は、一般に、ビデオデータ処理に関し、より詳細には、ビデオデータの変換スキップ残差符号化に関する。

【背景技術】

【０００３】

背景

［0003］ビデオの圧縮及び解凍の業界では、ビデオ符号化の新しい標準が開発されている。例えば、ＩＴＵ－Ｔビデオコーディングエキスパートグループ（Video Coding Expert Group、「ＶＣＥＧ」）と、ＩＳＯ／ＩＥＣムービングピクチャエキスパートグループ（Moving Picture Expert Group、「ＭＰＥＧ」）とによるジョイントビデオエキスパートチーム（Joint Video Experts Team、「ＪＶＥＴ」）が、バーサタイルビデオコーディング（Versatile Video Coding、「ＶＶＣ」）標準を現在開発している。ＶＶＣ標準は、その前身である高効率ビデオコーディング（High Efficiency Video Coding、「ＨＥＶＣ／Ｈ．２６５」）標準の圧縮効率を倍増することを目的としている。換言すれば、ＶＶＣの目標は、ＨＥＶＣ／Ｈ．２６５と同じ主観的な品質を半分の帯域幅で実現することである。

【発明の概要】

【課題を解決するための手段】

【０００４】

開示の概要

［0004］本開示の実施形態は、変換スキップ残差ビデオデータ符号化のための方法及びシステムを提供する。

【０００５】

［0005］１つの例示的な方法は、ビデオフレームのサブブロックの変換係数をスキャンする第１のパスを実行することを含み、スキャンする第１のパスが、変換係数のパリティレベルフラグをバイパス符号化することを含み、パリティレベルフラグが変換係数のレベルの絶対値のパリティを示す。

【０００６】

［0006］別の例示的な方法は、ビデオフレームのサブブロックの変換係数をスキャンする第１のパスを実行することを含み、スキャンする第１のパスが、変換係数のパリティレベルフラグをバイパス復号することを含み、パリティレベルフラグが変換係数のレベルの絶対値のパリティを示す。

【０００７】

［0007］１つの例示的なシステムは、一組の命令を格納するメモリと、プロセッサとを備え、プロセッサは、一組の命令を実行して、ビデオフレームのサブブロックの変換係数

をスキャンする第１のパスを実行することをシステムに行わせるように構成され、スキャンする第１のパスが、変換係数のパリティレベルフラグをバイパス符号化することを含み、パリティレベルフラグが変換係数のレベルの絶対値のパリティを示す。

【０００８】

［0008］ 別の例示的なシステムは、一組の命令を格納するメモリと、プロセッサとを備え、プロセッサは、一組の命令を実行して、ビデオフレームのサブブロックの変換係数をスキャンする第１のパスを実行することをシステムに行わせるように構成され、スキャンする第１のパスが、変換係数のパリティレベルフラグをバイパス復号することを含み、パリティレベルフラグが変換係数のレベルの絶対値のパリティを示す。

【０００９】

図面の簡単な説明

［0009］ 以下の詳細な説明及び添付の図面において、本開示の実施形態及び様々な態様を説明する。各図に示す様々な特徴は、必ずしも原寸に比例して描かれていない。

【図面の簡単な説明】

【００１０】

【図１】[0010]ハイブリッドビデオ符号化システムの例示的なエンコーダのブロック図を示す。

【図２】[0011]ハイブリッドビデオ符号化システムの例示的なデコーダのブロック図を示す。

【図３】[0012]変換符号化のシンタックスを含む例示的な擬似コードを示す。

【図４】[0013]変換スキップ残差符号化のシンタックスを含む例示的な擬似コードを示す。

【図５】[0014]本開示のいくつかの実施形態による、符号化パスの数を３に減らした、変換スキップ残差符号化の例示的な方法を示す。

【図６】[0015]本開示のいくつかの実施形態による、図５に示す方法のシンタックスを含む例示的な擬似コードを示す。

【図７】[0016]本開示のいくつかの実施形態による、符号化パスの数を３に減らした、変換スキップ残差符号化の別の例示的な方法を示す。

【図８】[0017]本開示のいくつかの実施形態による、図７に示す方法のシンタックスを含む例示的な擬似コードを示す。

【図９】[0018]本開示のいくつかの実施形態による、８×８の変換スキップブロックの例示的な逆方向スキャンを示す。

【図１０Ａ】[0019]本開示のいくつかの実施形態による、反転される前の８×８のブロックを示す。

【図１０Ｂ】[0020]本開示のいくつかの実施形態による、図１０Ａの８×８のブロックが反転された後の結果として得られるブロックを示す。

【図１１】[0021]本開示のいくつかの実施形態による、例示的なマルチパス符号化を示す。

【図１２】[0022]本開示のいくつかの実施形態による、レベルの絶対値の例示的なシングルパス符号化方法を示す。

【図１３】[0023]本開示のいくつかの実施形態による、ライスパラメータの例示的なルックアップテーブルを示す。

【図１４】[0024]本開示のいくつかの実施形態による、シングルパスバイパス符号化と組み合わされた、符号化パスの数を３に減らした、変換スキップ残差符号化の例示的な方法を示す。

【図１５】[0025]本開示のいくつかの実施形態による、図１４の方法と組み合わされた、バイパス符号化のシンタックスを含む例示的な擬似コードを示す。

【図１６】[0026]本開示のいくつかの実施形態による、第１のパスがコンテキスト符号化に対応し、第２のパスがゴロム・ライス符号化に対応する、変換スキップ残差符号化の例示的な方法を示す。

【図１７】[0027]本開示のいくつかの実施形態による、図１６の方法のバイパス符号化のシンタックスを含む例示的な擬似コードを示す。
【図１８】[0028]本開示のいくつかの実施形態による、最小バイパス符号化値が０に等しい場合のライスパラメータの例示的なルックアップテーブルを示す。
【発明を実施するための形態】
【００１１】
詳細な説明
［0029］ ここで、例示的な実施形態を詳細に参照し、例示的な実施形態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、添付の図面においては、別段の記載がない限り、異なる図面で同じ番号がある場合、この番号は同じ又は類似の要素を表している。例示的な実施形態の以下の説明に記載されている実装形態は、本発明と一致するすべての実装形態を表しているわけではない。そうではなく、これら実装形態は、添付の特許請求の範囲に記載されている本発明に関連する態様と一致する装置及び方法の例に過ぎない。以下に、本開示の特定の態様をより詳細に説明する。本明細書に記載されている用語及び定義が、参照により組み込まれている用語及び／又は定義と矛盾する場合には、本明細書の記載が優先される。
【００１２】
［0030］ ビデオは、視覚情報を格納するために時系列に配置された一組の静止画像（又は「フレーム」）である。これらの画像を時系列でキャプチャ及び格納するためには、ビデオキャプチャデバイス（例えば、カメラ）を使用することができ、そのような画像を時系列で表示するためには、ビデオ再生デバイス（例えば、テレビ、コンピュータ、スマートフォン、タブレットコンピュータ、ビデオプレーヤ、又は表示機能を備えた任意のエンドユーザ端末）を使用することができる。また、いくつかの用途では、ビデオキャプチャデバイスは、調査、会議、又は生放送などのために、キャプチャしたビデオをビデオ再生デバイス（例えば、モニタを備えたコンピュータ）にリアルタイムで送信することができる。
【００１３】
［0031］ このような用途で必要とされるストレージ空間及び伝送帯域幅を減らすために、ビデオを圧縮することができる。例えば、ビデオは、格納及び送信の前に圧縮することができ、表示の前に解凍することができる。圧縮及び解凍は、プロセッサ（例えば、汎用コンピュータのプロセッサ）によって実行されるソフトウェア又は専用ハードウェアによって実施することができる。圧縮のためのモジュールは一般に「エンコーダ」と呼ばれ、解凍のためのモジュールは一般に「デコーダ」と呼ばれる。エンコーダ及びデコーダは、「コーデック」と総称され得る。エンコーダ及びデコーダは、様々な適切なハードウェア、ソフトウェア、又はそれらの組み合わせのいずれかとして実装することができる。例えば、エンコーダ及びデコーダのハードウェア実装形態としては、１つ又は複数のマイクロプロセッサ、デジタルシグナルプロセッサ（ＤＳＰ）、特定用途向け集積回路（ＡＳＩＣ）、フィールドプログラマブルゲートアレイ（ＦＰＧＡ）、離散ロジック、又はそれらの任意の組み合わせなどの回路が挙げられる。エンコーダ及びデコーダのソフトウェア実装形態としては、プログラムコード、コンピュータ実行可能命令、ファームウェア、又はコンピュータ可読媒体に固定された任意の適切なコンピュータ実装アルゴリズム若しくはプロセスが挙げられる。ビデオの圧縮及び解凍は、ＭＰＥＧ－１、ＭＰＥＧ－２、ＭＰＥＧ－４、又はＨ．２６ｘシリーズなどの様々なアルゴリズム又は標準によって実装することができる。いくつかの用途では、コーデックは、第１の符号化標準からのビデオを解凍し、第２の符号化標準を使用して、解凍されたビデオを再圧縮することができ、この場合、コーデックは「トランスコーダ」と呼ばれ得る。
【００１４】
［0032］ ビデオ符号化プロセスは、画像を再構成するために使用され得る有用な情報を識別して保持することができる。ビデオ符号化プロセスで無視された情報を完全に再構成できない場合、符号化プロセスは「非可逆」と呼ばれ得る。それ以外の場合、符号化プロ

セスは「可逆」と呼ばれ得る。ほとんどの符号化プロセスは非可逆であり、このことは、必要なストレージ空間及び伝送帯域幅を減らすこととのトレードオフである。

【００１５】

[0033] 多くの場合、符号化される画像（「対象の画像」と呼ばれる）の有用な情報は、参照画像（例えば、以前に符号化又は再構成された画像）に対する変化を含み得る。このような変化としては、ピクセルの位置の変化、輝度の変化、又は色の変化が挙げられ、その中でも位置の変化が主に関係する。物体を表す一群のピクセルの位置の変化は、参照画像と対象の画像との間の物体の動きを反映し得る。

【００１６】

[0034] ＨＥＶＣ／Ｈ．２６５と同じ主観的な品質を半分の帯域幅を用いて実現するために、ＪＶＥＴは、ジョイントエクスプロレーションモデル（joint exploration model、「ＪＥＭ」）参照ソフトウェアを使用してＨＥＶＣを超える技術を開発してきた。符号化技術がＪＥＭに組み込まれたため、ＪＥＭはＨＥＶＣよりも大幅に高い符号化パフォーマンスを実現した。ＶＣＥＧ及びＭＰＥＧもまた、ＨＥＶＣを超える次世代ビデオ圧縮標準の開発を正式に開始した。

【００１７】

[0035] ＶＶＣ標準は、より優れた圧縮パフォーマンスを提供する多くの符号化技術を含めることを続けている。ＶＶＣは、ＨＥＶＣ、Ｈ．２６４／ＡＶＣ、ＭＰＥＧ２、Ｈ．２６３などの最新のビデオ圧縮標準で使用されてきたものと同じハイブリッドビデオ符号化システムに基づいている。図１は、ハイブリッドビデオ符号化システムの例示的なエンコーダのブロック図を示す。図１に示すように、ビデオエンコーダ２００は、ビデオブロック、又はビデオブロックのパーティション若しくはサブパーティションを含む、ビデオフレーム内のブロックのイントラ符号化又はインター符号化を実行し得る。イントラ符号化は、所与のビデオフレーム内のビデオにおける空間的冗長性を低減又は除去するために空間的予測に依拠し得る。インター符号化は、ビデオシーケンスの隣接するフレーム内のビデオにおける時間的冗長性を低減又は除去するために時間的予測に依拠し得る。イントラモードとは、いくつかの空間ベースの圧縮モードを指し得る。インターモード（片方向予測又は双方向予測など）は、いくつかの時間ベースの圧縮モードを指し得る。

【００１８】

[0036] 図１を参照すると、入力ビデオ信号２０２が、ブロックごとに処理され得る。例えば、ビデオブロックユニットは、１６×１６のピクセルブロック（例えば、マクロブロック（ＭＢ））であり得る。ビデオブロックユニットのサイズは、使用する符号化技法、並びに必要な精度及び効率に応じて異なり得る。ＨＥＶＣでは、拡張ブロックサイズ（例えば、符号化ツリーユニット（ＣＴＵ））を使用して、例えば１０８０ｐ以上の解像度のビデオ信号を圧縮することができる。ＨＥＶＣでは、ＣＴＵは、最大６４×６４のルーマサンプル、対応するクロマサンプル、及び関連するシンタックス要素を含み得る。ＶＶＣでは、ＣＴＵのサイズは更に増えて、１２８×１２８のルーマサンプル、対応するクロマサンプル、及び関連するシンタックス要素を含み得る。ＣＴＵは、例えば、四分木、二分木、又は三分木を使用して、符号化ユニット（ＣＵ）に更に分割され得る。ＣＵは、別個の予測方法が適用され得る予測ユニット（ＰＵ）に更に分割され得る。各入力ビデオブロックは、空間的予測ユニット２６０又は時間的予測ユニット２６２を使用することによって処理され得る。

【００１９】

[0037] 空間的予測ユニット２６０は、対象のブロックを含む同じ画像／スライスに関する情報を用いて、対象のブロック／ＣＵに対して空間的予測（例えば、イントラ予測）を実行する。空間的予測は、同じビデオ画像フレーム／スライス内の既に符号化された隣接ブロックからのピクセルを使用して、対象のビデオブロックを予測し得る。空間的予測は、ビデオ信号に固有の空間的冗長性を低減し得る。

【００２０】

[0038] 時間的予測ユニット２６２は、対象のブロックを含む画像／スライスとは異な

る画像／スライスからの情報を用いて、対象のブロックに対して時間的予測（例えば、インター予測）を実行する。ビデオブロックの時間的予測は、１つ又は複数の動きベクトルによって信号化され得る。片方向時間的予測では、対象のブロックの予測信号を生成するために、１つの参照画像を示す動きベクトルは１つしか使用されない。一方、双方向時間的予測では、それぞれの参照画像をそれぞれ示す２つの動きベクトルが、対象のブロックの予測信号を生成するために使用され得る。動きベクトルは、対象のブロックと参照フレーム内の１つ又は複数の関連するブロックとの間の動きの量及び方向を示し得る。複数の参照画像がサポートされる場合、１つ又は複数の参照画像インデックスがビデオブロックに送信され得る。１つ又は複数の参照画像インデックスは、時間的予測信号が参照画像ストア又は復号された画像バッファ（ＤＰＢ）２６４内のどの参照画像から到来し得るかを識別するために使用され得る。

【００２１】

［0039］ エンコーダのモード決定及びエンコーダ制御ユニット２８０は、例えば、レート歪み最適化に基づいて予測モードを選択し得る。決定された予測モードに基づいて、予測ブロックが取得され得る。予測ブロックは、加算器２１６において、対象のビデオブロックから差し引かれ得る。予測残差は、変換ユニット２０４によって変換され、量子化ユニット２０６によって量子化され得る。量子化された残差係数は、逆量子化ユニット２１０において逆量子化され、逆変換ユニット２１２において逆変換されて、再構成された残差を形成し得る。再構成された残差は、加算器２２６において予測ブロックに追加されて、再構成されたビデオブロックを形成し得る。ループフィルタリングの前の再構築されたビデオブロックは、イントラ予測のための参照サンプルを提供するために使用され得る。

【００２２】

［0040］ 再構成されたビデオブロックは、ループフィルタ２６６においてループフィルタリングを通り得る。例えば、デブロッキングフィルタ、サンプルアダプティブオフセット（ＳＡＯ）、及び適応ループフィルタ（ＡＬＦ）などのループフィルタリングが適用され得る。ループフィルタリングの後の再構成されたブロックは、参照画像ストア２６４に格納されてよく、他のビデオブロックを符号化するためのインター予測参照サンプルを提供するために使用され得る。出力ビデオビットストリーム２２０を形成するために、符号化モード（例えば、インター又はイントラ）、予測モード情報、動き情報、及び量子化された残差係数がエントロピー符号化ユニット２０８に送信されて、ビットレートを更に減らした後に、データが圧縮され、パックされてビットストリーム２２０を形成し得る。

【００２３】

［0041］ 図２は、ハイブリッドビデオ符号化システムの例示的なデコーダのブロック図を示す。図２に示すように、ビデオビットストリーム３０２は、エントロピー復号ユニット３０８においてアンパック又はエントロピー復号され得る。符号化モード情報は、空間的予測ユニット３６０を選択するか時間的予測ユニット３６２を選択するかを判定するために使用され得る。予測モード情報は、予測ブロックを生成するために対応する予測ユニットに送信され得る。例えば、動き補償予測は、時間的予測ユニット３６２によって適用されて、時間的予測ブロックを形成し得る。

【００２４】

［0042］ 残差係数は、再構成された残差を取得するために逆量子化ユニット３１０及び逆変換ユニット３１２に送信され得る。予測ブロックと再構成された残差とは、３２６において一緒に加算されて、ループフィルタリングの前の再構成されたブロックを形成し得る。次いで、再構成されたブロックは、ループフィルタ３６６においてループフィルタリングを通り得る。例えば、デブロッキングフィルタ、ＳＡＯ、及びＡＬＦなどのループフィルタリングが適用され得る。次いで、ループフィルタリングの後の再構成されたブロックは、参照画像ストア３６４に格納され得る。参照画像ストア３６４内の再構成されたデータは、復号されたビデオ３２０を取得するために使用され得る、又は将来のビデオブロックを予測するために使用され得る。復号されたビデオ３２０は、システム１００（図１）において説明されるようなディスプレイデバイス１４６などのディスプレイデバイス上

に表示され得る。

【００２５】

[0043] ＶＶＣ（例えば、ＶＶＣ５）では、ブロックは変換係数のＭ行Ｎ列の配列であり得る。変換係数は、変換において特定の１次元又は２次元の周波数インデックスに関連付けられる、周波数領域にあると見なされるスカラー量であり得る。変換係数レベルは、配列TransCoeffLevel[x0][y0][cIdx][xC][yC]によって表され得る。配列インデックスｘ０、ｙ０は、画像の左上のルーマサンプルに対する、考慮される変換ブロックの左上ルーマサンプルのロケーション（ｘ０，ｙ０）を指定し得る。配列インデックスcIdxは、色成分のインジケータを指定し得る。配列インデックスｘＣ及びｙＣは、対象の変換ブロック内の変換係数ロケーション（ｘＣ，ｙＣ）を指定し得る。

【００２６】

[0044] ＶＶＣ（例えば、ＶＶＣ５）では、符号化ブロックの変換係数は、重複しない係数グループ（又はサブブロック）を使用して符号化される。各サブブロックについて、通常の（又はコンテキスト）符号化されたビンとバイパス符号化されたビンとが符号化順に分離される。例えば、最初にサブブロックのすべての通常の符号化されたビンが送信され、その後、バイパス符号化されたビンが送信される。サブブロックの変換係数レベルは、スキャン位置を３回パスする間に符号化される。変換係数レベルは、変換係数の値であり得る。コンテキスト符号化の場合、各ビンは、コンテキストによって選択される確率モデルを有し得る。コンテキストは、以前に符号化されたシンタックス要素を指し得る。バイパス符号化の場合、特定のビンが、符号化効率の損失がほとんどない状態で符号化プロセスを高速化するために選択され得る。バイパス符号化では、ビンは、設定された確率（例えば、０．５に等しい確率）で符号化され得る。

【００２７】

[0045] パス１では、有意フラグ（例えば、sig_coeff_flag）、１より大のフラグ（例えば、gt1_flag）、パリティフラグ（例えば、par_level_flag）、及び３より大のフラグ（例えば、gt3_flag）が順に符号化される。有意フラグが１に等しい場合、最初に１より大のフラグが符号化される。１より大のフラグは、絶対レベル（例えば、レベルの絶対値）が１より大であるか否かを指定する。１より大のフラグが１に等しい場合、パリティフラグ及び３より大のフラグが符号化される。パリティフラグは、絶対レベルから２を差し引いたパリティを指定する。３より大のフラグは、絶対レベルが３より大であるか否かを指定する。最後の通常の（例えば、コンテキスト）符号化された係数の位置は、変数firstPosMode1に格納され得る。

【００２８】

[0046] パス２（ａ）では、残りの絶対レベル（例えば、abs_remainder）の符号化が、係数グループの最初のスキャン位置から開始してfirstPosMode1の位置まで処理される。１より大のフラグが１に等しい位置のみが符号化される。非２値シンタックス要素は、ゴロム・ライス符号で２値化され、結果として得られるビンは、算術符号化エンジンのバイパスモードで符号化される。例えば、値２は、「００１」のゴロム・ライス符号を使用して表すことができ、「００１」の各ビットは、ビン（例えば、ビン０、ビン０、及びビン１）と呼ばれ得る。

【００２９】

[0047] パス２（ｂ）では、絶対レベル（dec_abs_level）の符号化が、第１のバイパス符号化位置（例えば、逆順から、firstPosMode1－１）から開始して係数グループの最後のスキャン位置まで処理され、ゴロム・ライス符号を使用して算術符号化エンジンのバイパスモードで完全に符号化される。

【００３０】

[0048] パス３では、符号フラグ（例えば、sign_flag）の符号化が、sig_coeff_flagが１に等しいすべてのスキャン位置に対して処理される。

【００３１】

[0049] ４×４のサブブロックの場合、３２個以下の通常の符号化されたビン（例えば

、sig_coeff_flag、gt1_flag、par_level_flag、及びgt3_flag）が符号化又は復号されると期待され得る。２×２のクロマサブブロックの場合、通常符号化されたビンの数は８に制限され得る。限度に到達した後、すべてのビンがバイパスモードで符号化される。

【００３２】

[0050] ＪＶＥＴが変換スキップ残差ブロックのために採用した新しい残差符号化プロセスでは、変換スキップ残差符号化の係数のスキャンの順序は順方向スキャンであり、変換スキップブロックの左上の位置から開始する。サブブロックの変換スキップ係数レベルは、スキャン位置を６回パスする間に符号化される。

【００３３】

[0051] パス１では、sig_coeff_flag、coeff_sign_flag、１より大のフラグ（例えば、abs_level_gtx_flag[0]）、及びpar_level_flagが符号化順に処理される。sig_coeff_flagが１に等しい場合、coeff_sign_flag及びabs_level_gtx_flag[0]が順に符号化される。coeff_sign_flagは、変換係数レベルの符号を指定する。abs_level_gtx_flag[0]は、絶対レベルが１より大であるか否かを指定する。abs_level_gtx_flag[0]が１に等しい場合、par_level_flagが追加的に符号化される。par_level_flagは、絶対レベルから２を差し引いたパリティを指定する。フラグを符号化する前には、コンテキスト適応型２値算術符号化（context-adaptive binary arithmetic coding、「ＣＡＢＡＣ」）エンジンが、コンテキスト符号化されたビンが利用可能か否かをチェックする。コンテキスト符号化されたビンが利用可能ではない場合、そのフラグはバイパス符号化される。

【００３４】

[0052] パス２では、所与の位置のabs_level_gtx_flag[0]が１に等しい場合、３より大のフラグ（例えば、abs_level_gtx_flag[1]）が符号化される。abs_level_gtx_flag[1]は、絶対レベルが３より大であるか否かを指定する。各係数のabs_level_gtx_flag[1]を符号化する前には、ＣＡＢＡＣエンジンが、コンテキスト符号化されたビンが利用可能か否かをチェックする。コンテキスト符号化されたビンが利用可能ではない場合、abs_level_gtx_flag[1]はバイパス符号化される。

【００３５】

[0053] パス３では、所与の位置のabs_level_gtx_flag[1]が１に等しい場合、５より大のフラグ（例えば、abs_level_gtx_flag[2]）が符号化される。abs_level_gtx_flag[2]は、絶対レベルが５より大であるか否かを指定する。各係数のabs_level_gtx_flag[2]を符号化する前には、ＣＡＢＡＣエンジンが、コンテキスト符号化されたビンが利用可能か否かをチェックする。コンテキスト符号化されたビンが利用可能ではない場合、abs_level_gtx_flag[2]はバイパス符号化される。

【００３６】

[0054] パス４では、所与の位置のabs_level_gtx_flag[2]が１に等しい場合、７より大のフラグ（例えば、abs_level_gtx_flag[3]）が符号化される。abs_level_gtx_flag[3]は、絶対レベルが７より大であるか否かを指定する。各係数のabs_level_gtx_flag[3]を符号化する前には、ＣＡＢＡＣエンジンが、コンテキスト符号化されたビンが利用可能か否かをチェックする。コンテキスト符号化されたビンが利用可能ではない場合、abs_level_gtx_flag[3]はバイパス符号化される。

【００３７】

[0055] パス５では、ある位置のabs_level_gtx_flag[3]が１に等しい場合、９より大のフラグ（例えば、abs_level_gtx_flag[4]）が符号化される。abs_level_gtx_flag[4]は、絶対レベルが９より大であるか否かを指定する。各係数のabs_level_gtx_flag[4]を符号化する前には、ＣＡＢＡＣエンジンが、コンテキスト符号化されたビンが利用可能か否かをチェックする。コンテキスト符号化されたビンが利用可能ではない場合、abs_level_gtx_flag[4]はバイパス符号化される。

【００３８】

[0056] パス６では、abs_remainderが、abs_level_gtx_flag[4]が１に等しいすべてのスキャン位置に対して処理される。非２値シンタックス要素は、ゴロム・ライス符号

で２値化され、結果として得られるビンは、算術符号化エンジンのバイパスモードで符号化される。

【００３９】

［0057］図３は、変換符号化のシンタックスを含む例示的な擬似コードを示す。例えば、図３に示すシンタックスは、ＶＶＣにおける変換符号化に使用され得る。図４は、変換スキップ残差符号化のシンタックスを含む例示的な擬似コードを示す。例えば、図４に示すシンタックスは、ＶＶＣにおける変換スキップ残差符号化に使用され得る。

【００４０】

［0058］変換スキップ残差符号化の現在の設計にはいくつかの問題がある。第１に、変換スキップ残差符号化の符号化パスの数は６回である。つまり、多くの場合、ＣＡＢＡＣエンジンは係数グループを６回スキャンする必要があり、このことはＣＡＢＡＣのスループットに大きく影響する。第２に、変換スキップ残差符号化の符号化パスの数は、変換残差符号化の符号化パスの数とは異なる（例えば、６回のパス対３回のパス）。符号化パスの数の差により、ハードウェアの実装が複雑になり得る。第３に、変換スキップ残差符号化の係数スキャンは順方向スキャンであるのに対し、変換残差符号化のスキャンは逆順である。スキャンの順序の違いによっても、ハードウェアの実装が複雑になり得る。第４に、変換残差符号化では、バイパス符号化は、２つのシンタックス要素（例えば、abs_remainder及びdec_abs_level）しか持たない。しかしながら、変換スキップ残差符号化では、バイパス符号化は、sig_coeff_flag、coeff_sign_flag、par_level_flag、abs_level_gtx_flag[0]、abs_level_gtx_flag[1]、abs_level_gtx_flag[2]、abs_level_gtx_flag[3]、abs_level_gtx_flag[4]、及びabs_remainderなどの更に多くのシンタックス要素を有し得る。変換残差符号化及び変換スキップ残差符号化のバイパス符号化方法を統合することが望ましい。

【００４１】

［0059］本開示の実施形態は、パリティフラグの新しいバイパス符号化方法を提供する。パリティフラグは、絶対レベルから２を差し引いたパリティを指定し、これは、非ゼロ係数位置の絶対値が偶数であるか奇数であるかを示す。変換スキップブロックにおけるパリティフラグの確率分布は一様であり得る。したがって、偶数値の確率と奇数値の確率とは等しくなり得る。これにより、パリティフラグ及び信号パリティフラグのバイパス符号化が適用され得る。このバイパス符号化は、abs_remainderが信号化される前又は後に行われ得る。パリティフラグは、１より大のフラグが１に等しい場合に信号化され得る。いくつかの実施形態では、コンテキスト符号化されたビンの総数は、変換ブロック内のサンプル当たり２つのビンに制限され得るため、パリティフラグのコンテキストを解放することにより、等しくない確率分布を有する他のシンタックス要素（例えば、パリティフラグの後に符号化されたｘより大のフラグのいずれか）をコンテキスト符号化させることができ、これにより符号化効率を向上させることができる。

【００４２】

［0060］いくつかの実施形態では、パリティフラグは、通常のコンテキスト符号化されたビンとして符号化され、係数符号化プロセスの第２のパスで信号化され得る。パリティフラグは、３より大のフラグの前又は後に信号化され得る。

【００４３】

［0061］ＶＶＣ（例えば、ＶＶＣ５）では、全部で４つの符号化パスが、３より大のフラグ（例えば、abs_level_gtx_flag[1]）、５より大のフラグ（例えば、abs_level_gtx_flag[2]）、７より大のフラグ（abs_level_gtx_flag[3]）、及び９より大のフラグ（例えば、abs_level_gtx_flag[4]）を符号化するために使用される。換言すれば、各フラグは別個の符号化パスで符号化される。本開示の実施形態は、単一のパスですべてのフラグを符号化するための新しい方法を提供する。その結果、すべてのフラグを符号化するために必要な符号化パスは１つだけであり、これにより、ＣＡＢＡＣのスループットを向上させることができる。

【００４４】

[0062] いくつかの実施形態では、３より大のフラグ（例えば、abs_level_gtx_flag[1]）の位置は、第１の符号化パスに移動され得る。例えば、１より大のフラグ（例えば、abs_level_gtx_flag[1]）が１に等しい場合、３より大のフラグが信号化され得る。この変更により、変換残差符号化のケース１より大のフラグ及び３より大のフラグの符号化を統合することができ、またこれにより、第１の符号化パスでこれら２つのフラグが符号化される。

【００４５】

[0063] いくつかの実施形態では、サブブロックの変換スキップレベルの符号化パスの数が、３に減らされ得る。図５は、本開示のいくつかの実施形態による、符号化パスの数を３に減らした、変換スキップ残差符号化の例示的な方法を示す。図５の方法は３つのパスを含む。

【００４６】

[0064] パス１（ステップ５０２）では、サブブロックの係数がスキャンされる。いくつかの実施形態では、サブブロックの最初のスキャン位置から開始してそのサブブロックの最後のスキャン位置までの各係数がスキャンされる。各係数には有意フラグ（例えば、sig_coeff_flag）が存在し得る。いくつかの実施形態では、有意フラグは、係数のレベルが非ゼロ値であるか否かを指定し得る。係数の有意フラグが、レベルが非ゼロ値であることを示している（例えば、有意フラグが１に等しい）場合、信号係数符号フラグ（例えば、coeff_sign_flag）及び１より大のフラグ（例えば、abs_level_gtx_flag[0]）が符号化され得る。１より大のフラグは、レベルの絶対値が１より大であるか否かを指定し得る。

【００４７】

[0065] パス２（ステップ５０４）では、サブブロックの係数がスキャンされる。いくつかの実施形態では、サブブロックの最初のスキャン位置から開始してそのサブブロックの最後のスキャン位置までの各係数がスキャンされる。係数の１より大のフラグが、絶対レベルが１より大であることを示している（例えば、１より大のフラグが１に等しい）場合、３より大のフラグ（例えば、abs_level_gtx_flag[1]）が符号化され得る。３より大のフラグは、絶対レベルが３より大であるか否かを指定し得る。係数の３より大のフラグが、絶対レベルが３より大であることを示している（例えば、３より大のフラグが１に等しい）場合、５より大のフラグ（例えば、abs_level_gtx_flag[2]）が符号化され得る。５より大のフラグは、絶対レベルが５より大であるか否かを指定し得る。係数の５より大のフラグが、絶対レベルが５より大であることを示している（例えば、５より大のフラグが１に等しい）場合、７より大のフラグ（例えば、abs_level_gtx_flag[3]）が符号化され得る。７より大のフラグは、絶対レベルが７より大であるか否かを指定し得る。係数の７より大のフラグが、絶対レベルが７より大であることを示している（例えば、７より大のフラグが１に等しい）場合、９より大のフラグ（例えば、abs_level_gtx_flag[4]）が符号化され得る。９より大のフラグは、絶対レベルが９より大であるか否かを指定し得る。

【００４８】

[0066] パス３（ステップ５０６）では、サブブロックの係数がスキャンされる。いくつかの実施形態では、サブブロックの最初のスキャン位置から開始してそのサブブロックの最後のスキャン位置までの各係数がスキャンされる。係数の１より大のフラグが、絶対レベルが１より大であることを示している（例えば、１より大のフラグが１に等しい）場合、係数のパリティレベルフラグ（例えば、par_level_flag）がバイパス符号化され得る。パリティレベルフラグは、絶対レベルから２を差し引いたパリティを指定し得る。９より大のフラグが、絶対レベルが９より大であることを示している（例えば、９より大のフラグが１に等しい）場合、係数の残りの絶対レベル（例えば、abs_remainder）が符号化され得、非２値シンタックス要素がゴロム・ライス符号で２値化され得る。いくつかの実施形態では、結果として得られるビンが、算術符号化エンジンのバイパスモードで符号化され得る。

【００４９】

[0067] 図６は、本開示のいくつかの実施形態による、図５に示す方法のシンタックス

を含む例示的な擬似コードを示す。図６の擬似コードの一部はイタリック体で示され、３より大のフラグ、５より大のフラグ、７より大のフラグ、９より大のフラグ、及びパリティレベルフラグの処理を示している。

【００５０】

[0068] 図５の方法は、エンコーダ（例えば、図１のエンコーダ）によって実施され得ることを理解されたい。いくつかの実施形態では、エンコーダはビデオフレームを受信し得る。デコーダ（例えば、図２のデコーダ）については、本方法は３つのパスを含み得る。

【００５１】

[0069] パス１では、サブブロックの係数がスキャンされる。いくつかの実施形態では、サブブロックの最初のスキャン位置から開始してそのサブブロックの最後のスキャン位置までの各係数がスキャンされる。各係数には有意フラグ（例えば、sig_coeff_flag）が存在し得る。有意フラグは復号され得る。有意フラグは、レベルが非ゼロ値であるか否かを指定し得る。係数の有意フラグが、レベルが非ゼロ値であることを示している（例えば、有意フラグが１に等しい）場合、信号係数符号フラグ（例えば、coeff_sign_flag）及び１より大のフラグ（例えば、abs_level_gtx_flag[0]）が復号され得る。

【００５２】

[0070] パス２では、サブブロックの係数がスキャンされる。いくつかの実施形態では、サブブロックの最初のスキャン位置から開始してそのサブブロックの最後のスキャン位置までの各係数がスキャンされる。係数の１より大のフラグが、絶対レベルが１より大であることを示している（例えば、１より大のフラグが１に等しい）場合、各係数のパリティレベルフラグ（例えば、par_level_flag）が復号され得る。パリティレベルフラグは、絶対レベルから２を差し引いたパリティを指定し得る。係数の１より大のフラグが、絶対レベルが１より大であることを示している（例えば、１より大のフラグが１に等しい）場合、３より大のフラグ（例えば、abs_level_gtx_flag[1]）が復号され得る。係数の３より大のフラグが、絶対レベルが３より大であることを示している（例えば、３より大のフラグが１に等しい）場合、５より大のフラグ（例えば、abs_level_gtx_flag[2]）が復号され得る。係数の５より大のフラグが、絶対レベルが５より大であることを示している（例えば、４より大のフラグが１に等しい）場合、７より大のフラグ（例えば、abs_level_gtx_flag[3]）が復号され得る。係数の７より大のフラグが、絶対レベルが７より大であることを示している（例えば、７より大のフラグが１に等しい）場合、９より大のフラグ（例えば、abs_level_gtx_flag[4]）が復号され得る。

【００５３】

[0071] パス３では、サブブロックの係数がスキャンされる。いくつかの実施形態では、サブブロックの最初のスキャン位置から開始してそのサブブロックの最後のスキャン位置までの各係数がスキャンされる。９より大のフラグが、絶対レベルが９より大であることを示している（例えば、９より大のフラグが１に等しい）場合、係数の残りの絶対レベル（例えば、abs_remainder）が復号され得る。

【００５４】

[0072] いくつかの実施形態では、デコーダは、ビデオビットストリームを受信し得る。

【００５５】

[0073] 図７は、本開示のいくつかの実施形態による、符号化パスの数を３に減らした、変換スキップ残差符号化の別の例示的な方法を示す。図７の方法は３つのパスを含み得る。

【００５６】

[0074] パス１（ステップ７０２）では、サブブロックの係数がスキャンされる。いくつかの実施形態では、サブブロックの最初のスキャン位置から開始してそのサブブロックの最後のスキャン位置までの各係数がスキャンされる。各係数には有意フラグ（例えば、sig_coeff_flag）が存在し得る。有意フラグは、レベルが非ゼロ値であるか否かを示し得る。係数の有意フラグが、レベルが非ゼロ値であることを示している（例えば、有意フラグが１に等しい）場合、信号係数符号フラグ（例えば、coeff_sign_flag）及び１より大

のフラグ（例えば、abs_level_gtx_flag[0]）が符号化され得る。１より大のフラグは、レベルの絶対値が１より大であるか否かを指定し得る。係数の１より大のフラグが、レベルの絶対値が１より大であることを示している（例えば、１より大のフラグが１に等しい）場合、３より大のフラグ（例えば、abs_level_gtx_flag[1]）が符号化され得る。３より大のフラグは、絶対レベルが３より大であるか否かを指定し得る。

【００５７】

[0075] パス２（ステップ７０４）では、サブブロックの係数がスキャンされる。いくつかの実施形態では、サブブロックの最初のスキャン位置から開始してそのサブブロックの最後のスキャン位置までの各係数がスキャンされる。係数の３より大のフラグが、絶対レベルが３より大であることを示している（例えば、３より大のフラグが１に等しい）場合、５より大のフラグ（例えば、abs_level_gtx_flag[2]）が符号化され得る。５より大のフラグは、絶対レベルが５より大であるか否かを指定し得る。係数の５より大のフラグが、絶対レベルが５より大であることを示している（例えば、５より大のフラグが１に等しい）場合、７より大のフラグ（例えば、abs_level_gtx_flag[3]）が符号化され得る。７より大のフラグは、絶対レベルが７より大であるか否かを指定し得る。係数の７より大のフラグが、絶対レベルが７より大であることを示している（例えば、７より大のフラグが１に等しい）場合、９より大のフラグ（例えば、abs_level_gtx_flag[4]）が符号化され得る。９より大のフラグは、絶対レベルが９より大であるか否かを指定し得る。

【００５８】

[0076] パス３（ステップ７０６）では、サブブロックの係数がスキャンされる。いくつかの実施形態では、サブブロックの最初のスキャン位置から開始してそのサブブロックの最後のスキャン位置までの各係数がスキャンされる。係数の１より大のフラグが、絶対レベルが１より大であることを示している（例えば、１より大のフラグが１に等しい）場合、係数のパリティレベルフラグ（例えば、par_level_flag）がバイパス符号化され得る。パリティレベルフラグは、絶対レベルから２を差し引いたパリティを指定し得る。９より大のフラグが、絶対レベルが９より大であることを示している（例えば、９より大のフラグが１に等しい）場合、係数の残りの絶対レベル（例えば、abs_remainder）が処理され得、非２値シンタックス要素がゴロム・ライス符号で２値化され得る。結果として得られるビンは、算術符号化エンジンのバイパスモードで符号化され得る。

【００５９】

[0077] 図５の方法と比較すると、図７の方法は、パス２ではなくパス１で３より大のフラグの符号化を処理する。図８は、本開示のいくつかの実施形態による、図７に示す方法のシンタックスを含む例示的な擬似コードを示す。図８の擬似コードの一部はイタリック体で示され、３より大のフラグ、５より大のフラグ、７より大のフラグ、９より大のフラグ、及びパリティレベルフラグの処理を示している。

【００６０】

[0078] 図７の方法は、エンコーダ（例えば、図１のエンコーダ）によって実施され得ることを理解されたい。いくつかの実施形態では、エンコーダはビデオフレームを受信し得る。デコーダ（例えば、図２のデコーダ）については、本方法は３つのパスを含み得る。

【００６１】

[0079] パス１では、サブブロックの係数がスキャンされる。いくつかの実施形態では、サブブロックの最初のスキャン位置から開始してそのサブブロックの最後のスキャン位置までの各係数がスキャンされる。各係数には有意フラグ（例えば、sig_coeff_flag）が存在し得る。有意フラグは復号される。有意フラグは、レベルが非ゼロ値であるか否かを示し得る。係数の有意フラグが、レベルが非ゼロ値であることを示している（例えば、有意フラグが１に等しい）場合、信号係数符号フラグ（例えば、coeff_sign_flag）及び１より大のフラグ（例えば、abs_level_gtx_flag[0]）が復号され得る。係数の１より大のフラグが、絶対レベルが１より大であることを示している（例えば、１より大のフラグが１に等しい）場合、３より大のフラグ（例えば、abs_level_gtx_flag[1]）が復号され得る。

【００６２】
［0080］ パス２では、サブブロックの係数がスキャンされる。いくつかの実施形態では、サブブロックの最初のスキャン位置から開始してそのサブブロックの最後のスキャン位置までの各係数がスキャンされる。係数の１より大のフラグが、絶対レベルが１より大であることを示している（例えば、１より大のフラグが１に等しい）場合、各係数のパリティレベルフラグ（例えば、par_level_flag）が復号され得る。係数の３より大のフラグが、絶対レベルが３より大であることを示している（例えば、３より大のフラグが１に等しい）場合、５より大のフラグ（例えば、abs_level_gtx_flag[2]）が復号され得る。係数の５より大のフラグが、絶対レベルが５より大であることを示している（例えば、５より大のフラグが１に等しい）場合、７より大のフラグ（例えば、abs_level_gtx_flag[3]）が復号され得る。係数の７より大のフラグが、絶対レベルが７より大であることを示している（例えば、７より大のフラグが１に等しい）場合、９より大のフラグ（例えば、abs_level_gtx_flag[4]）が復号され得る。９より大のフラグは、絶対レベルが９より大であるか否かを指定し得る。
【００６３】
［0081］ パス３では、サブブロックの係数がスキャンされる。いくつかの実施形態では、サブブロックの最初のスキャン位置から開始してそのサブブロックの最後のスキャン位置までの各係数がスキャンされる。９より大のフラグが、絶対レベルが９より大であることを示している（例えば、９より大のフラグが１に等しい）場合、係数の残りの絶対レベル（例えば、abs_remainder）が復号され得る。
【００６４】
［0082］ いくつかの実施形態では、デコーダは、ビデオビットストリームを受信し得る。
【００６５】
［0083］ いくつかの実施形態では、変換残差符号化プロセスと変換スキップ残差符号化プロセスとのスキャンの順序を統合するために、変換スキップ残差ブロックのスキャンの順序が変更され得る。例えば、変換スキップ残差ブロックのスキャンの順序は、順方向スキャンから逆方向スキャンに変更され得る。図９は、本開示のいくつかの実施形態による、８×８の変換スキップブロックの例示的な逆方向スキャンを示す。図９に示すように、係数のスキャンは、右下隅にある係数から開始され、左上隅にある係数で終了する。図９に示すスキャンの順序は、図５及び図７に示す方法によって適用され得ることを理解されたい。
【００６６】
［0084］ いくつかの実施形態では、逆方向スキャンは、変換ブロックが反転された後に実行され得る。図１０Ａ及び図１０Ｂは、本開示のいくつかの実施形態による、８×８のブロックの例示的な反転を示す。図１０Ａ及び図１０Ｂに示すように、ブロックを反転した後、左上の残差係数の位置が右下の位置に移動される。ブロックが反転した後、逆方向スキャン（例えば、図９の逆方向スキャン）が実行され得る。
【００６７】
［0085］ いくつかの実施形態では、バイパス符号化は、複数のパスで実行され得る。図１１は、本開示のいくつかの実施形態による、例示的なマルチパス符号化を示す。いくつかの実施形態では、図１１に示すマルチパス符号化は、ＶＶＣ（例えば、ＶＶＣ５）で実行され得る。図１１のコンテキスト符号化されたビンの数は、（図１１の黒色の点として示されている）第１のパスの位置において最大限度に到達したと想定され得る。図１１に示すように、フラグ（例えば、３より大のフラグ、５より大のフラグ、７より大のフラグ、９より大のフラグなど）は、複数の符号化パスでバイパス符号化され得る。
【００６８】
［0086］ いくつかの実施形態では、レベルの絶対値のシングルパスバイパス符号化が実現され得る。図１２は、本開示のいくつかの実施形態による、レベルの絶対値の例示的なシングルパス符号化方法を示す。図１２に示すように、コンテキスト符号化されたビンが（例えば、図１２に黒色の点として示されている）最大限度に到達すると、ＣＡＢＡＣエ

ンジンは、ゴロム・ライス符号化を使用して絶対レベルの残りのバイパス符号化を開始し得る。

【００６９】

[0087] 図１３は、本開示のいくつかの実施形態による、ライスパラメータの例示的なルックアップテーブルを示す。図１３によれば、ライスパラメータは変数cRiceParamで表され得る。絶対和のロケーションは、変数locSumAbsで表され得る。

【００７０】

[0088] いくつかの実施形態では、ライスパラメータ（例えば、cRiceParam）は、以下のようにして導出され得る。変換スキップブロックの配列AbsLevel[x][y]、左上のルーマロケーション（ｘ０，ｙ０）、及び対象の係数スキャンロケーション（ｘＣ，ｙＣ）を所与として、minLevelが最小バイパス符号化値であると仮定すると、係数のフラグのいずれもコンテキスト符号化されない場合、その係数のminLevelは０である。フラグのすべてがコンテキスト符号化されている場合、minLevelは１０に等しい。

【００７１】

[0089] いくつかの実施形態では、変数locSumAbsは、以下の擬似コードによって指定されるように導出され得る。

```
locSumAbs = 0
if (xC 〉 0)
  locSumAbs += AbsLevel[xC - 1][yC]
if (yC 〉 0)
  locSumAbs += AbsLevel[xC][yC -1]
locSumAbs += ( 10 - minLevel)
locSumAbs = Clip3 (0, 31, LocSumAbs)
```

【００７２】

[0090] いくつかの実施形態では、シングルパスバイパス符号化は、図７に示す方法と組み合わされ得る。例えば、図１４は、本開示のいくつかの実施形態による、シングルパスバイパス符号化と組み合わされた、符号化パスの数を３に減らした、変換スキップ残差符号化の例示的な方法を示す。図１４の方法は３つのパスを含む。

【００７３】

[0091] パス１（ステップ１４０２）では、サブブロックの係数がスキャンされる。いくつかの実施形態では、サブブロックの最初のスキャン位置から開始して最後のスキャン位置までの各係数がスキャンされる。各係数について、コンテキスト符号化されたビンの残りの数がグループ限度以上である場合、以下が実行され得る。例えば、いくつかの実施形態では、コンテキスト符号化又はバイパス符号化のいずれかを使用して、グループ内のフラグのすべてを符号化することがより効率的である。その結果、コンテキスト符号化されたビンの数がグループ限度より少ない場合は、残りのフラグの一部をコンテキスト符号化で符号化し、他のフラグをバイパス符号化で符号化するよりも、残りのフラグのすべてをバイパス符号化で符号化する方が効率的である。この例では、パス１のグループにおいて符号化された４つのフラグ（例えば、有意フラグ、信号係数符号フラグ、１より大のフラグ、及び３より大のフラグ）が存在し得る。したがって、４のグループ限度を所与として、各係数について、コンテキスト符号化されたビンの残りの数が４以上である場合、以下が実行され得る。有意フラグ（例えば、sig_coeff_flag）が、レベルが非ゼロ値であることを示している（例えば、有意フラグが１に等しい）場合、信号係数符号フラグ（例えば、coeff_sign_flag）及び１より大のフラグ（例えば、abs_level_gtx_flag[0]）が符号化され得る。１より大のフラグが、絶対レベルが１より大であることを示している（例えば、１より大のフラグが１に等しい）場合、３より大のフラグ（例えば、abs_level_gtx_flag[1]）が符号化され得る。いくつかの実施形態では、係数は、コンテキスト符号化されたビンが最大限度（例えば、図１２の黒色の点）に到達するまでスキャンされ得、その後、スキャンは、第１のパスの最後の位置としての位置で停止する。

【００７４】

［0092］パス１の終了後、且つパス２の開始前に、以前のパスの最後の位置にしたがって第１のパスバイパス位置変数（例えば、iFirstPassBypassPos）が設定され得る（ステップ１４０４）。いくつかの実施形態では、第１のパスバイパス位置変数は、以前のパスに１を加えた最後の位置に設定され得る。

【００７５】

［0093］パス２（ステップ１４０６）では、サブブロックの最初のスキャン位置から開始してiFirstPassBypassPosまでの係数がスキャンされ得る。各係数について、コンテキスト符号化されたビンの残りの数がグループ限度以上である場合、以下が実行され得る。例えば、グループ限度は３とすることができる（例えば、５より大のフラグ、７より大のフラグ、及び９より大のフラグ）。３より大のフラグが、絶対レベルが３より大であることを示している（例えば、３より大のフラグが１に等しい）場合、５より大のフラグ（例えば、abs_level_gtx_flag[2]）が符号化され得る。５より大のフラグが、絶対レベルが５より大であることを示している（例えば、５より大のフラグが１に等しい）場合、７より大のフラグ（例えば、abs_level_gtx_flag[3]）が符号化され得る。７より大のフラグが、絶対レベルが７より大であることを示している（例えば、７より大のフラグが１に等しい）場合、９より大のフラグ（例えば、abs_level_gtx_flag[4]）が符号化され得る。

【００７６】

［0094］パス２の終了後、且つパス３の開始前に、以前のパスの最後の位置にしたがって第２のパスバイパス位置変数（例えば、iSecondPassBypassPos）が設定される（ステップ１４０８）。いくつかの実施形態では、第２のパスバイパス位置変数は、以前のパスに１を加えた最後の位置に設定され得る。

【００７７】

［0095］パス３（ａ）（ステップ１４１０）では、サブブロックの最初のスキャン位置から開始してiFirstPassBypassPosまでの係数がスキャンされ得る。各係数について以下が実行され得る。１より大のフラグが、絶対レベルが１より大であることを示している（例えば、１より大のフラグが１に等しい）場合、パリティレベルフラグ（例えば、par_level_flag）がバイパス符号化され得る。９より大のフラグが、絶対レベルが９より大であることを示している（例えば、９より大のフラグが１に等しい）場合、残りの絶対レベル（例えば、abs_remainder）の符号化は、ゴロム・ライス符号で２値化される非２値シンタックス要素によって処理され得る。結果として得られるビンは、算術符号化エンジンのバイパスモードで符号化され得る。

【００７８】

［0096］パス３（ｂ）（ステップ１４１２）では、iFirstPassBypassPosから開始して、最後のスキャン位置までの係数がスキャンされ得る。各係数について、絶対レベル（例えば、dec_abs_level）の符号化が、ゴロム・ライス符号で２値化された非２値シンタックス要素によって処理され得る。結果として得られるビンは、算術符号化エンジンのバイパスモードで符号化され得る。信号係数符号フラグも符号化され得る。

【００７９】

［0097］いくつかの実施形態では、パス３（ｂ）では、iFirstPassBypassPosから開始して、最後のスキャン位置までの係数がスキャンされ得、絶対レベル（例えば、dec_abs_level）の符号化が、各係数について２ステップで処理され得る。第１に、dec_abs_levelがゼロであるか否かが信号化される。dec_abs_levelが非ゼロである場合、ゴロム・ライス符号で２値化される非２値シンタックス要素、及び結果として得られるビンは、算術符号化エンジンのバイパスモードで符号化され得る。信号係数符号フラグも符号化され得る。

【００８０】

［0098］図１５は、本開示のいくつかの実施形態による、図１４の方法と組み合わされた、バイパス符号化のシンタックスを含む例示的な擬似コードを示す。図１５の擬似コードの一部はイタリック体で示され、３より大のフラグ、５より大のフラグ、７より大のフラグ、９より大のフラグ、及びパリティレベルフラグの処理を示している。

【００８１】
［0099］いくつかの実施形態では、２パス符号化方法が使用され得る。例えば、パス１ではフラグのすべてがコンテキスト符号化され得、パス２ではバイパス符号化にゴロム・ライス符号化が使用され得る。図１６は、本開示のいくつかの実施形態による、第１のパスがコンテキスト符号化に対応し、第２のパスがゴロム・ライス符号化に対応する、変換スキップ残差符号化の例示的な方法を示す。図１６の方法は２つのパスを含む。
【００８２】
［00100］パス１（ステップ１６０２）では、サブブロックの係数がスキャンされる。いくつかの実施形態では、サブブロックの最初のスキャン位置から開始して最後のスキャン位置までの各係数がスキャンされる。各係数について、コンテキスト符号化されたビンの残りの数が８以上である場合、以下が実行され得る。有意フラグ（例えば、sig_coeff_flag）がコンテキスト符号化され得る。有意フラグが、レベルが非ゼロであることを示している（例えば、有意フラグが１に等しい）場合、係数符号フラグ（例えば、coeff_sign_flag）及び１より大のフラグ（例えば、abs_level_gtx_flag[0]）が信号化され得る。１より大のフラグが、絶対レベルが１より大であることを示している（例えば、１より大のフラグが１に等しい）場合、パリティレベルフラグ（例えば、par_level_flag）及び３より大のフラグ（例えば、abs_level_gtx_flag[1]）が符号化され得る。パリティレベルフラグは、絶対レベルから２を差し引いたパリティを指定し得る。３より大のフラグは、絶対レベルが３より大であるか否かを指定し得る。３より大のフラグが、絶対レベルが３より大であることを示している（例えば、３より大のフラグが１に等しい）場合、５より大のフラグ（例えば、abs_level_gtx_flag[2]）が符号化され得る。５より大のフラグが、絶対レベルが５より大であることを示している（例えば、５より大のフラグが１に等しい）場合、７より大のフラグ（例えば、abs_level_gtx_flag[3]）が符号化され得る。７より大のフラグが、絶対レベルが７より大であることを示している（例えば、７より大のフラグが１に等しい）場合、９より大のフラグ（例えば、abs_level_gtx_flag[4]）が符号化され得る。９より大のフラグは、レベルの絶対レベルが９より大であるか否かを指定し得る。いくつかの実施形態では、係数は、コンテキスト符号化されたビンが最大限度（例えば、図１２の黒色の点）に到達するまでスキャンされ得、その後、スキャンは、第１のパスの最後の位置としての位置で停止する。
【００８３】
［00101］パス１の終了後、且つパス２の開始前に、以前のパスの最後の位置に応じて第１のパスバイパス位置変数（iFirstPassBypassPos）が設定され得る（ステップ１６０４）。いくつかの実施形態では、第１のパスバイパス位置変数は、以前のパスに１を加えた最後の位置に設定され得る。第１のパスバイパス位置変数は、絶対レベル（例えば、dec_abs_level）シンタックスが信号化される開始位置を表し得る。スキャン位置が第１のパスバイパス位置よりも小さい係数は、パス１でコンテキスト符号化を介して部分的に信号化され得、残りの係数はパス２（ａ）で信号化され得る。いくつかの実施形態では、係数のスキャン位置が第１のパスバイパス位置変数以上である場合、その係数のフラグのいずれもコンテキスト符号化されず、完全な係数及び符号は、パス２（ｂ）のバイパス符号化を使用して信号化され得る。
【００８４】
［00102］パス２（ａ）（ステップ１６０６）では、サブブロックの最初のスキャン位置から開始して第１のパスバイパス位置変数から１を差し引いたロケーションまでの係数がスキャンされ得る。各係数について以下が実行され得る。９より大のフラグが、絶対レベルが９より大であることを示している（例えば、９より大のフラグが１に等しい）場合、残りの絶対レベル（例えば、abs_remainder）がゴロム・ライス符号を使用して２値化され得、結果として得られるビンが算術符号化エンジンのバイパスモードで符号化され得る。
【００８５】
［00103］パス２（ｂ）（ステップ１６０８）では、第１のパスバイパス位置から開始

して、最後のスキャン位置までの係数がスキャンされ得る。各係数について以下が実行され得る。絶対レベル（例えば、dec_abs_level）は、ゴロム・ライス符号で２値化され得、結果として得られるビンは、算術符号化エンジンのバイパスモードで符号化され得る。絶対レベルが０に等しくない場合、係数符号フラグ（例えば、coeff_sign_flag）はバイパス符号化され得る。

【００８６】

[00104] 図１６の方法は、エンコーダ（例えば、図１のエンコーダ）によって実施され得ることを理解されたい。いくつかの実施形態では、エンコーダはビデオフレームを受信し得る。図１７の方法からの符号化されたビデオフレームは、デコーダ（例えば、図２のデコーダ）を使用して復号され得ることを理解されたい。いくつかの実施形態では、復号方法は２つのパスを含み得る。

【００８７】

[00105] パス１では、サブブロックの係数がスキャンされる。いくつかの実施形態では、サブブロックの最初のスキャン位置から開始して最後のスキャン位置までの各係数がスキャンされる。各係数について、コンテキスト符号化されたビンの残りの数がグループ限度以上である場合、以下が実行され得る。例えば、グループ限度は８とすることができ、パス１で符号化される様々なフラグ（例えば、有意フラグ、係数符号フラグ、１より大のフラグ、パリティレベルフラグ、３より大のフラグ、５より大のフラグ、７より大のフラグ、及び９より大のフラグ）の数を示す。有意フラグ（例えば、sig_coeff_flag）がコンテキスト復号され得る。有意フラグが、レベルが非ゼロであることを示している（例えば、有意フラグが１に等しい）場合、係数符号フラグ（例えば、coeff_sign_flag）及び１より大のフラグ（例えば、abs_level_gtx_flag[0]）が信号化され得る。１より大のフラグが、絶対レベルが１より大であることを示している（例えば、１より大のフラグが１に等しい）場合、パリティレベルフラグ（例えば、par_level_flag）及び３より大のフラグ（例えば、abs_level_gtx_flag[1]）が復号され得る。パリティレベルフラグは、絶対レベルから２を差し引いたパリティを指定し得る。３より大のフラグは、絶対レベルが３より大であるか否かを指定し得る。３より大のフラグが、絶対レベルが３より大であることを示している（例えば、３より大のフラグが１に等しい）場合、５より大のフラグ（例えば、abs_level_gtx_flag[2]）が復号され得る。５より大のフラグが、絶対レベルが５より大であることを示している（例えば、５より大のフラグが１に等しい）場合、７より大のフラグ（例えば、abs_level_gtx_flag[3]）が復号され得る。７より大のフラグが、絶対レベルが７より大であることを示している（例えば、７より大のフラグが１に等しい）場合、９より大のフラグ（例えば、abs_level_gtx_flag[4]）が復号され得る。９より大のフラグは、レベルの絶対レベルが９より大であるか否かを指定し得る。いくつかの実施形態では、係数は、コンテキスト符号化されたビンが最大限度（例えば、図１２の黒色の点）に到達するまでスキャンされ得、その後、スキャンは、第１のパスの最後の位置としての位置で停止する。

【００８８】

[00106] パス１の終了後、且つパス２の開始前に、以前のパスの最後の位置に応じて第１のパスバイパス位置変数（iFirstPassBypassPos）が設定され得る。いくつかの実施形態では、第１のパスバイパス位置変数は、以前のパスに１を加えた最後の位置に設定され得る。第１のパスバイパス位置変数は、絶対レベル（例えば、dec_abs_level）シンタックスが信号化される開始位置を表し得る。スキャン位置が第１のパスバイパス位置よりも小さい係数は、パス１でコンテキスト符号化を介して部分的に信号化され得、残りの係数はパス２（ａ）で信号化され得る。いくつかの実施形態では、係数のスキャン位置が第１のパスバイパス位置変数以上である場合、その係数のフラグのいずれもコンテキスト復号されず、完全な係数及び符号は、パス２（ｂ）のバイパス復号を使用して信号化され得る。

【００８９】

[00107] パス２（ａ）では、サブブロックの最初のスキャン位置から開始して第１の

パスバイパス位置変数から１を差し引いたロケーションまでの係数がスキャンされ得る。各係数について以下が実行され得る。９より大のフラグが、絶対レベルが９より大であることを示している（例えば、９より大のフラグが１に等しい）場合、２値化された残りの絶対レベル（例えば、abs_remainder）がゴロム・ライス符号を使用して復号され得、結果として得られるビンが算術符号化エンジンのバイパスモードで復号され得る。

【００９０】

[00108] パス２（ｂ）では、第１のパスバイパス位置から開始して、最後のスキャン位置までの係数がスキャンされ得る。各係数について以下が実行され得る。２値化された絶対レベル（例えば、dec_abs_level）は、ゴロム・ライス符号で復号され得、結果として得られるビンは、算術符号化エンジンのバイパスモードで復号され得る。絶対レベルが０に等しくない場合、係数符号フラグ（例えば、coeff_sign_flag）はバイパス復号され得る。

【００９１】

[00109] 図１７は、本開示のいくつかの実施形態による、図１６の方法のバイパス符号化のシンタックスを含む例示的な擬似コードを示す。図１７の擬似コードの一部はイタリック体で示され、第１のパスバイパス位置とパス２（ｂ）との処理を示している。

【００９２】

[00110] いくつかの実施形態では、図１７に示すように、パス１は、コンテキスト符号化されたビンの残りの数が８以上（例えば、図１７に示すように「MaxCcbs 〉=8」）である場合にのみ実行される。このことは、コンテキスト符号化ビンバジェットで最大７つのコンテキスト符号化が「浪費」され得、符号化のパフォーマンスに悪影響を与え得ることを意味する。したがって、いくつかの実施形態では、abs_level_gtx_flag[]フラグの数は、開示された２パス符号化方法で調整され得る。例えば、９より大のフラグ（例えば、abs_level_gtx_flag[4]）を符号化する代わりに、７より大のフラグ（例えば、abs_level_gtx_flag[3]））までしか符号化しない。したがって、コンテキスト符号化されたビンの残りの数が７以上である場合にのみ、パス１が実行され得る。いくつかの実施形態では、５より大のフラグ（abs_level_gtx_flag[2]）までしか符号化されない。したがって、コンテキスト符号化されたビンの残りの数が６以上である場合にのみ、パス１が実行される。abs_level_gtx_flag[]フラグの数は任意の数に調整され得ることを理解されたい。abs_level_gtx_flag[]フラグの数を調整することにより、第１の符号化パスでより多くの位置を符号化することができるため、より良好な符号化効率が提供される。

【００９３】

[00111] いくつかの実施形態では、ライスパラメータcRiceParamは、以下のようにして導出され得る。スキャンロケーション（ｘＣ，ｙＣ）における係数値としての配列TransCoeffLevel[xC][yC]、及び最小バイパス符号化値としての値minLevelを所与として、係数のフラグのいずれもコンテキスト符号化されない場合、その係数のminLevelは０である。フラグのすべてがコンテキスト符号化されている場合、minLevelは１０に等しい。変数locSumAbsは、以下の擬似コードによって指定されるように導出され得る。

```
locSumAbs = 0
if (xC 〉 0)
  locSumAbs += Abs(TransCoeffLevel[xC - 1][yC])
if (yC 〉 0)
  locSumAbs += Abs(TransCoeffLevel[xC][yC -1])
locSumAbs = locSumAbs - 2*minLevel
locSumAbs = Clip3 (0, 31, LocSumAbs)
```

【００９４】

[00112] 図１８は、本開示のいくつかの実施形態による、最小バイパス符号化値が０に等しい場合のライスパラメータの例示的なルックアップテーブルを示す。図１８に示すように、ライスパラメータは変数cRiceParamで表され得る。絶対和のロケーションは、変数locSumAbsで表され得る。

【００９５】
［00113］ いくつかの実施形態では、２つの別個のルックアップテーブルが、minLevelに基づくライスパラメータの導出に使用され得る。minLevelが０に等しい場合、図１８に示すテーブルが使用され得る。minLevelが０に等しくない場合、図１３に示すテーブルが使用され得る。
【００９６】
［00114］ いくつかの実施形態では、ライスパラメータの導出は、ルックアップテーブルを必要としない。例えば、cRiceParamは次のように導出され得る。

```
cRiceParam = (locSumAbs + offset) 〉〉 3
```

【００９７】
［00115］ 上記の式では、オフセットは事前定義された定数であり得、オフラインイントレーニングによって決定され得る。オフセット値の一例は４である。
【００９８】
［00116］ いくつかの実施形態では、オフセット値は色成分に依存する。例えば、オフセット値は、ルーマの場合は４、クロマの場合は０であり得る。
【００９９】
［00117］ いくつかの実施形態では、オフセット値はフレームタイプに依存する。例えば、オフセット値は、イントラフレームの場合は４、インターフレームの場合は０であり得る。
【０１００】
［00118］ いくつかの実施形態では、ＶＶＣ（例えば、ＶＶＣ７）では、変換スキップモードは、ルーマ成分とクロマ成分との両方に対して可能であり、両方のタイプの成分が同じコンテキスト変数を共有し得る。コンテキスト変数は、最近復号されたビンを含む式によって、ビンの適応型２値算術復号プロセスに指定された変数であり得る。しかしながら、ルーマブロック及びクロマブロックの信号統計は異なり得る。結果として、本開示のいくつかの実施形態では、異なるコンテキスト変数がルーマ成分及びクロマ成分に使用され得る。提案されたコンテキストモデルの拡張によって影響を受けるシンタックス要素としては、有意係数フラグ（例えば、sig_coeff_flag）、abs_level_gtx_flag[n][j]（例えば、ｊ＝０～４）、パリティフラグ（例えば、par_level_flag）、信号係数符号フラグ（例えば、coeff_sign_flag）、及びcoded_sub_block_flag（例えば、図３に示すcoded_sub_block_flag）が挙げられる。
【０１０１】
［00119］ ＶＶＣ（例えば、ＶＶＣ７）では、３つのコンテキスト変数が、変換スキップモードのsig_coeff_flagを符号化するために使用され得る。いくつかの実施形態では、合計６つのコンテキスト変数（例えば、ルーマに３つ、クロマに３つ）が、変換スキップモードのsig_coeff_flagを符号化するために使用され得る。変換スキップモードのsig_coeff_flagを符号化するためのコンテキストインデックスは、隣（例えば、上隣と左隣）の有意係数の数から導出され得る。いくつかの実施形態では、コンテキストインデックスは、コンテキスト変数の識別子を指し得る。例えば、利用可能なコンテキスト変数が６つある場合、第１のコンテキスト変数のコンテキストインデックスは０、第２のコンテキスト変数のコンテキストインデックスは１などになり得る。このプロセスへの入力は、色成分インデックスcIdx、ルーマロケーション（ｘ０，ｙ０）、及び対象の係数スキャンロケーション（ｘＣ，ｙＣ）であり得る。ルーマロケーション（ｘ０，ｙ０）は、対象の画像の特定のサンプル（例えば、左上サンプル）に対して、対象の変換ブロックの特定のサンプル（例えば、左上サンプル）を指定し得る。このプロセスの出力は、符号化インデックス変数ctxIncであり得る。いくつかの実施形態では、変数ctxIncは、以下の擬似コードにしたがって導出され得る。

```
locNumSig = 0
if (xC 〉 0)
  locNumSig += sig_coeff_flag[xC - 1][yC]
```

```
  if (yC 〉 0)
    locNumSig += sig_coeff_flag[xC][yC - 1]
  if (cIdx == 0) // 例えば、ルーマ成分では、ctxIncは次のように導出され得る。
    ctxInc = locNumSig
  else // 例えば、クロマ成分では、ctxIncは次のように導出され得る。
    ctxInc = locNumSig + 3
```

【０１０２】

[00120] ＶＶＣ（例えば、ＶＶＣ７）では、４つのコンテキスト変数が、変換スキップモードのabs_level_gtx_flag[n][0]を符号化するために使用され得る。いくつかの実施形態では、合計８つのコンテキスト変数（ルーマに４つ及びクロマに４つ）が、変換スキップモードのabs_level_gtx_flag[n][0]を符号化するために使用され得る。変換スキップのabs_level_gtx_flag[n][0]を符号化するためのコンテキストインデックスは、隣（例えば、上隣と左隣）の有意係数の数から導出され得る。このプロセスへの入力は、色成分インデックスcIdx、ルーマロケーション（ｘ０，ｙ０）、及び対象の係数スキャンロケーション（ｘＣ，ｙＣ）であり得る。ルーマロケーション（ｘ０，ｙ０）は、対象の画像の特定のサンプル（例えば、左上サンプル）に対して、対象の変換ブロックの特定のサンプル（例えば、左上サンプル）を指定し得る。このプロセスの出力は、符号化インデックス変数ctxIncであり得る。いくつかの実施形態では、変数ctxIncは、以下の擬似コードにしたがって導出され得る。

```
  if (BdpcmFlag[x0][y0][cIdx] == 1)
    ctxInc = 3
  else if (xC 〉 0 and yC 〉 0)
    ctxInc = sig_coeff_flag[xC - 1][yC] + sig_coeff_flag[xC][yC - 1]
  else if (xC 〉 0)
    ctxInc = sig_coeff_flag[xC - 1][yC]
  else if (yC 〉 0)
    ctxInc = sig_coeff_flag[xC][yC - 1]
  else
    ctxInc = 0
  if (cIdx 〉 0)
    ctxInc = ctxInc + 4
```

【０１０３】

[00121] ＶＶＣ（例えば、ＶＶＣ７）では、１つのコンテキスト変数が、変換スキップモードのpar_level_flagを符号化するために使用され得る。いくつかの実施形態では、合計２つのコンテキスト変数（ルーマに１つ、クロマに１つ）が、変換スキップモードのpar_level_flagを符号化するために使用され得る。変換スキップのpar_level_flagを符号化するためのコンテキストインデックスは、次のように導出され得る。このプロセスへの入力は、色成分インデックスcIdxであり得る。このプロセスの出力は、符号化インデックス変数ctxIncである。いくつかの実施形態では、変数ctxIncは、以下の擬似コードにしたがって導出され得る。

```
  if (cIdx == 0)
    ctxInc = 0
  else
    ctxInc = 1
```

【０１０４】

[00122] いくつかの実施形態では、ルーマとクロマとに別個のコンテキスト変数が、変換スキップモードのabs_level_gtx_flagを符号化するために使用され得る。変換スキップのabs_level_gtx_flagを符号化するためのコンテキストインデックスは、次のように導出され得る。このプロセスへの入力は、色成分インデックスcIdxであり得る。このプロセスの出力は、符号化インデックス変数ctxIncであり得る。いくつかの実施形態では、変数

ctxIncは、以下のように導出され得る。シンタックス要素abs_level_gtx_flag[n][j]（j〉 0）のコンテキストインデックスが、ctxInc = j - 1として導出され、クロマ成分（例えば、0より大であるcIdx）については、コンテキストインデックスは、ctxInc = ctxInc + 4のようにインクリメントされる。

【０１０５】

[00123] ＶＶＣ（例えば、ＶＶＣ７）では、６つのコンテキスト変数が、変換スキップモードのcoeff_sign_flagを符号化するために使用され得る。いくつかの実施形態では、合計１２のコンテキスト変数（ルーマに６つ、クロマに６つ）が、変換スキップモードのcoeff_sign_flagを符号化するために使用され得る。変換スキップモードのcoeff_sign_flagを符号化するためのコンテキストインデックスは、隣（例えば、上隣と左隣）のcoeff_sign_flagから導出され得る。このプロセスへの入力は、色成分インデックスcIdx、ルーマロケーション（ｘ０，ｙ０）、及び対象の係数スキャンロケーション（ｘＣ，ｙＣ）であり得る。ルーマロケーション（ｘ０，ｙ０）は、対象の画像の特定のサンプル（例えば、左上サンプル）に対して、対象の変換ブロックの特定のサンプル（例えば、左上サンプル）を指定し得る。このプロセスの出力は、符号化インデックス変数ctxIncであり得る。いくつかの実施形態では、変数leftSign及びaboveSignは、以下の擬似コードにしたがって導出され得る。

```
if (xC == 0)
  leftSign = 0
else
  leftSign = CoeffSignLevel[xC - 1][yC]
if (yC == 0)
  aboveSign = 0
else
  aboveSign = CoeffSignLevel[xC][yC - 1]
```

【０１０６】

[00124] いくつかの実施形態では、変数ctxIncは、以下の擬似コードにしたがって導出され得る。

```
if ((leftSign == 0 and aboveSign == 0) or (leftSign == aboveSign))
  if (BdpcmFlag[x0][y0][cIdx] == 0)
    ctxInc = 0
  else
    ctxInc = 3
else if (leftSign 〉= 0 and aboveSign 〉= 0)
  if (BdpcmFlag[x0][y0][cIdx])
    ctxInc = 1
  else
    ctxInc = 4
else
  if (BdpcmFlag[x0][y0][cIdx])
    ctxInc = 2
  else
    ctxInc = 5
if (cIdx 〉 0)
  ctxInc = ctxInc + 6
```

【０１０７】

[00125] ＶＶＣ（例えば、ＶＶＣ７）では、３つのコンテキスト変数が、変換スキップモードのcoded_sub_block_flagを符号化するために使用され得る。coded_sub_block_flagは、サブブロックの変換係数レベルが０に等しいか否かを指定するサブブロックフラグであり得る。例えば、coded_sub_block_flag[xS][yS]が０に等しい場合、ロケーシ

ョン（ｘＳ，ｙＳ）におけるサブブロックの変換係数レベルは０に等しいと推測される。coded_sub_block_flag[xS][yS]が１に等しい場合、ロケーション（ｘＳ，ｙＳ）におけるサブブロックの変換係数レベルの少なくとも１つが非ゼロ値を有する。いくつかの実施形態では、合計６つのコンテキスト変数（ルーマに３つ、クロマに３つ）が、変換スキップモードのcoded_sub_block_flagを符号化するために使用され得る。変換スキップモードのcoded_sub_block_flagを符号化するためのコンテキストインデックスは、上隣と左隣のcoded_sub_block_flagから導出され得る。このプロセスへの入力は、色成分インデックスcIdx、ルーマロケーション（ｘ０，ｙ０）、対象のサブブロックスキャンロケーション（ｘＳ，ｙＳ）、シンタックス要素coded_sub_block_flagの以前に復号されたビン、変換ブロック幅の２進対数log2TbWidth、及び変換ブロック高さlog2TbHeightであり得る。ルーマロケーション（ｘ０，ｙ０）は、対象の画像の左上サンプルに対して、対象の変換ブロックの左上サンプルを指定し得る。このプロセスの出力は、符号化インデックス変数ctxIncであり得る。いくつかの実施形態では、変数log2SbWidth及びlog2SbHeightは、以下の擬似コードにしたがって導出され得る。

```
if (minimum (log2TbWidth, log2TbHeight) 〈 2)
  log2SbWidth = 1
else
  log2SbHeight = log2SbWidth
```

【０１０８】

[00126] いくつかの実施形態では、変数log2SbWidth及びlog2SbHeightは、以下の擬似コードにしたがって修正され得る。

```
if (log2TbWidth 〈 2 and cIdx == 0)
  log2SbWidth = log2TbWidth
  log2SbHeight = 4 - log2SbWidth
else if (log2TbHeight 〈 2 and cIdx == 0)
  log2SbHeight = log2TbHeight
  log2SbWidth = 4 - log2SbHeight
```

【０１０９】

[00127] いくつかの実施形態では、変数csbfCtxは、０として初期化され、以下の擬似コードにしたがって修正され得る。

```
if (xS 〉 0)
  csbfCtx = csbfCtx + coded_sub_block_flag[xS - 1][yS]
if (yS 〉 0)
  csbfCtx = csbfCtx + coded_sub_block_flag[xS][yS - 1]
```

【０１１０】

[00128] いくつかの実施形態では、コンテキストインデックス変数ctxIncは、以下の擬似コードにしたがって、色成分インデックスcIdx及びcsbfCtxを使用して導出され得る。

```
if (cIdx == 0)
  ctxInc = csbfCtx
else
  ctxInc = 3 + csbfCtx
```

【０１１１】

[00129] いくつかの実施形態では、命令を含む非一時的コンピュータ可読記憶媒体も提供され、命令は、上記の方法を実行するためのデバイス（開示されているエンコーダ及びデコーダなど）によって実行され得る。非一時的媒体の一般的な形態としては、例えば、フロッピー（登録商標）ディスク、フレキシブルディスク、ハードディスク、ソリッドステートドライブ、磁気テープ、又は任意の他の磁気データ記憶媒体、ＣＤ－ＲＯＭ、任意の他の光学データ記憶媒体、孔のパターンを持つ任意の物理媒体、ＲＡＭ、ＰＲＯＭ、及びＥＰＲＯＭ、ＦＬＡＳＨ（登録商標）－ＥＰＲＯＭ又は任意の他のフラッシュメモリ、ＮＶＲＡＭ、キャッシュ、レジスタ、任意の他のメモリチップ又はカートリッジ、並び

にそれらのネットワークバージョンが挙げられる。デバイスは、１つ又は複数のプロセッサ（ＣＰＵ）、入出力インターフェース、ネットワークインターフェース、及び／又はメモリを備え得る。

【０１１２】

[00130] 本明細書において、「第１」及び「第２」などの関係用語は、エンティティ又は動作を別のエンティティ又は動作から区別するためにのみ使用され、これらのエンティティ又は動作の間のいかなる実際の関係又は順序をも必要としたり示唆したりするものではないことに留意されたい。更に、「含む／備える（comprising）」、「有する（having）」、「含む（containing）」、及び「含む／備える（including）」の単語、並びに他の同様の形態は、意味において均等であり、これらの単語のいずれか１つに続く１つ又は複数の項目が、そのような１つ若しくは複数の項目の網羅的な列挙であることを意味しない、又は列挙された１つ若しくは複数の項目のみに限定されることを意味しないという点でオープンエンドであることが意図されている。

【０１１３】

[00131] 本明細書で使用される場合、特に別段の記載のない限り、「又は」という用語は、実行不可能な場合を除いて、すべての可能な組み合わせを包含する。例えば、データベースがＡ又はＢを含み得ると記載されている場合、特に別段の記載のない限り、又は実行不可能でない限り、データベースは、Ａ、又はＢ、又はＡ及びＢを含み得る。別の例として、データベースがＡ、Ｂ、又はＣを含み得ると記載されている場合、特に別段の記載のない限り、又は実行不可能でない限り、データベースは、Ａ、又はＢ、又はＣ、又はＡ及びＢ、又はＡ及びＣ、又はＢ及びＣ、又はＡ及びＢ及びＣを含み得る。

【０１１４】

[00132] 上記の実施形態は、ハードウェア、又はソフトウェア（プログラムコード）、又はハードウェアとソフトウェアとの組み合わせによって実装され得ることを理解されたい。ソフトウェアによって実装される場合、ソフトウェアは上記のコンピュータ可読媒体に格納され得る。ソフトウェアは、プロセッサによって実行されると、開示された方法を実行し得る。本開示で説明されるコンピューティングユニット及び他の機能ユニットは、ハードウェア、又はソフトウェア、又はハードウェアとソフトウェアとの組み合わせによって実装され得る。また、上記のモジュール／ユニットのうちの複数が１つのモジュール／ユニットとして組み合わされてもよく、上記のモジュール／ユニットのそれぞれが複数のサブモジュール／サブユニットに更に分割されてもよいことが当業者には理解されよう。

【０１１５】

[00133] 上記明細書では、実装形態ごとに変化し得る多数の具体的な詳細を参照して実施形態を説明してきた。説明された実施形態に対して、ある程度の適合及び修正を行うことができる。当業者であれば、本明細書を検討すれば、また本明細書に開示される本発明を実施することにより、他の実施形態を想到し得る。本明細書及び各例は単なる例示として見なされ、本発明の真の範囲及び趣旨は以下の特許請求の範囲によって示されることが意図されている。また、図に示されている一連の工程は、説明のみを目的としており、いかなる特定の一連の工程にも限定されるように意図されていないことも意図されている。そのため、同じ方法を実施しながら、これらの工程を異なる順序で実行し得ることを当業者は理解されよう。

【０１１６】

[00134] 以下の条項を使用して実施形態を更に説明することができる。

１．

ビデオデータのエンコーダによって実施される符号化方法であって、

ビデオフレームのサブブロックの変換係数をスキャンする第１のパスを実行することを含み、

スキャンする第１のパスが、変換係数のパリティレベルフラグをバイパス符号化することを含み、パリティレベルフラグが変換係数のレベルの絶対値のパリティを示す、

符号化方法。
２．
スキャンする第１のパスの前に、変換係数をスキャンする第２のパスを実行することを更に含み、
スキャンする第２のパスが、１より大のフラグを符号化することを含み、１より大のフラグが、絶対値が１より大であるか否かを示し、
スキャンする第１のパスが、１より大のフラグが、絶対値が１より大であると示すことに応じて、パリティレベルフラグをバイパス符号化することを更に含む、
条項１に記載の符号化方法。
３．
サブブロックが複数の変換係数を有し、スキャンする第２のパスを実行することが、
コンテキスト符号化されたビンの数が最大限度に到達するまで、複数の変換係数をスキャンすることと、
コンテキスト符号化されたビンの数が最大限度に到達したことに応じて、第２のパスでスキャンされない変換係数のレベルの絶対値をバイパス符号化することであって、バイパス符号化することが、ゴロム・ライス符号化を使用してスキャンされない変換係数の絶対値を２値化することを含む、ことと
を更に含む、条項２に記載の符号化方法。
４．
スキャンする第２のパスが、
変換係数の有意フラグを符号化することであって、有意フラグが、変換係数のレベルがゼロであるか否かを示す、ことと、
有意フラグが、変換係数のレベルがゼロではないと示すことに応じて、１より大のフラグを符号化することと
を更に含む、条項２又は３に記載の符号化方法。
５．
スキャンする第２のパスが、
１より大のフラグが、絶対値が１より大であると示すことに応じて、変換係数の３より大のフラグを符号化することを更に含み、３より大のフラグが、絶対値が３より大であるか否かを示す、
条項２～４のいずれか一項に記載の符号化方法。
６．
スキャンする第２のパスの後、且つスキャンする第１のパスの前に、変換係数をスキャンする第３のパスを実行することを更に含み、
スキャンする第３のパスが、
３より大のフラグが、絶対値が３より大であると示すことに応じて、変換係数の５より大のフラグを符号化することであって、５より大のフラグが、絶対値が５より大であるか否かを示す、ことと、
５より大のフラグが、絶対値が５より大であると示すことに応じて、変換係数の７より大のフラグを符号化することであって、７より大のフラグが、絶対値が７より大であるか否かを示す、ことと、
７より大のフラグが、絶対値が７より大であると示すことに応じて、変換係数の９より大のフラグを符号化することであって、９より大のフラグが、絶対値が９より大であるか否かを示す、ことと
を含む、こと
条項５に記載の符号化方法。
７．
スキャンする第１のパスが、
９より大のフラグが、絶対値が９より大であると示すことに応じて、変換係数の残りの絶対レベルフラグを符号化することを更に含み、残りの絶対レベルフラグが、変換係数

のレベルの残りの絶対値を示す、
条項６に記載の符号化方法。
８．
スキャンする第２のパスの後、且つスキャンする第１のパスの前に、変換係数をスキャンする第３のパスを実行することを更に含み、
スキャンする第３のパスが、
１より大のフラグが、絶対値が１より大であると示すことに応じて、変換係数の３より大のフラグを符号化することであって、３より大のフラグが、絶対値が３より大であるか否かを示す、こと
を更に含む、
条項２に記載の符号化方法。
９．
スキャンする第３のパスが、
３より大のフラグが、絶対値が３より大であると示すことに応じて、変換係数の５より大のフラグを符号化することであって、５より大のフラグが、絶対値が５より大であるか否かを示す、ことと、
５より大のフラグが、絶対値が５より大であると示すことに応じて、変換係数の７より大のフラグを符号化することであって、７より大のフラグが、絶対値が７より大であるか否かを示す、ことと、
７より大のフラグが、絶対値が７より大であると示すことに応じて、変換係数の９より大のフラグを符号化することであって、９より大のフラグが、絶対値が９より大であるか否かを示す、ことと
を更に含む、条項８に記載の符号化方法。
１０．
スキャンする第１のパスが、
９より大のフラグが、絶対値が９より大であると示すことに応じて、変換係数の残りの絶対レベルフラグを符号化することを更に含み、残りの絶対レベルフラグが、変換係数のレベルの残りの絶対値を示す、
条項９に記載の符号化方法。
１１．
サブブロックが複数の変換係数を有し、スキャンする第１のパスが、逆の順序で複数の変換係数をスキャンすることによって実行される、条項１～１０のいずれか一項に記載の符号化方法。
１２．
スキャンする第１のパスを実行することの前に、複数の変換係数を反転すること
を更に含む、条項１１に記載の符号化方法。
１３．
符号化方法が変換スキップ残差符号化方法である、条項１～１２のいずれか一項に記載の符号化方法。
１４．
ビデオフレームを受信することと、
ビデオフレームを複数のサブブロックに分割することと
を更に含む、条項１～１３のいずれか一項に記載の符号化方法。
１５．
ビデオデータのデコーダによって実施される復号方法であって、
ビデオフレームのサブブロックの変換係数をスキャンする第１のパスを実行することを含み、
スキャンする第１のパスが、変換係数のパリティレベルフラグをバイパス復号することを含み、パリティレベルフラグが変換係数のレベルの絶対値のパリティを示す、
復号方法。

１６．
スキャンする第１のパスの前に、変換係数をスキャンする第２のパスを実行することを更に含み、
スキャンする第２のパスが、１より大のフラグを復号することを含み、１より大のフラグが、絶対値が１より大であるか否かを示し、
スキャンする第１のパスが、１より大のフラグが、絶対値が１より大であると示すことに応じて、第１のパスにおいてパリティレベルフラグをバイパス復号することを更に含む、
条項１５に記載の復号方法。
１７．
スキャンする第２のパスが、
変換係数の有意フラグを復号することであって、有意フラグが、変換係数のレベルがゼロであるか否かを示す、ことと、
有意フラグが、変換係数のレベルがゼロではないと示すことに応じて、１より大のフラグを復号することと
を更に含む、条項１６に記載の復号方法。
１８．
スキャンする第１のパスが、
１より大のフラグが、絶対値が１より大であると示すことに応じて、変換係数の３より大のフラグを復号することであって、３より大のフラグが、絶対値が３より大であるか否かを示す、ことと、
３より大のフラグが、絶対値が３より大であると示すことに応じて、変換係数の５より大のフラグを復号することであって、５より大のフラグが、絶対値が５より大であるか否かを示す、ことと、
５より大のフラグが、絶対値が５より大であると示すことに応じて、変換係数の７より大のフラグを復号することであって、７より大のフラグが、絶対値が７より大であるか否かを示す、ことと、
７より大のフラグが、絶対値が７より大であると示すことに応じて、変換係数の９より大のフラグを復号することであって、９より大のフラグが、絶対値が９より大であるか否かを示す、ことと
を更に含む、条項１６に記載の復号方法。
１９．
変換係数をスキャンする第３のパスを実行することを更に含み、
スキャンする第３のパスが、９より大のフラグが、絶対値が９より大であると示すことに応じて、変換係数の残りの絶対レベルフラグを復号することを含み、残りの絶対レベルフラグが、変換係数のレベルの残りの絶対値を示す、
条項１８に記載の復号方法。
２０．
サブブロックが複数の変換係数を有し、
スキャンする第１のパスが、サブブロックの右下隅からサブブロックの左上隅まで複数の変換係数をスキャンすることによって実行される、条項１５に記載の復号方法。
２１．
スキャンする第１のパスを実行することの前に、複数の変換係数を反転すること
を更に含む、条項２０に記載の復号方法。
２２．
復号方法が変換スキップ残差復号方法である、条項１５～２１のいずれか一項に記載の復号方法。
２３．
ビデオデータを符号化するシステムであって、本システムが、
一組の命令を格納するメモリと、
プロセッサとを備え、プロセッサが、一組の命令を実行して、

ビデオフレームのサブブロックの変換係数をスキャンする第１のパスを実行することをシステムに行わせるように構成され、
スキャンする第１のパスが、変換係数のパリティレベルフラグをバイパス符号化することを含み、パリティレベルフラグが変換係数のレベルの絶対値のパリティを示す、
ビデオデータを符号化するシステム。
２４．
ビデオデータを復号するシステムであって、本システムが、
一組の命令を格納するメモリと、
プロセッサとを備え、プロセッサが、一組の命令を実行して、
ビデオフレームのサブブロックの変換係数をスキャンする第１のパスを実行することをシステムに行わせるように構成され、
スキャンする第１のパスが、変換のパリティレベルフラグを復号することでを含み、パリティレベルフラグが変換係数のレベルの絶対値のパリティを示す、
ビデオデータを復号するシステム。
２５．
ビデオデータのエンコーダによって実施される符号化方法であって、
ビデオフレームのサブブロックの変換係数をスキャンする第１のパスを実行することと、
ブブロックの変換係数をスキャンする第２のパスを実行することとを含み、
スキャンする第１のパスを実行することが、コンテキスト符号化されたビンの数が最大限度に到達すると停止され、
サブブロックの変換係数の第１のセットが第１のパスでスキャンされ、
スキャンする第１のパスが、変換係数の第１のセットのそれぞれについて、変換係数のレベルがゼロであるか否かを示す有意フラグを符号化することを含み、
スキャンする第２のパスが、変換係数の第２のセットのそれぞれについてレベルの絶対値を２値化することを含み、変換係数の第２のセットが第１のパスでスキャンされない、
符号化方法。
２６．
スキャンする第１のパスが、
第１の変換係数の有意フラグが、第１の変換係数のレベルがゼロではないと示すことに応じて、第１の変換係数の１より大のフラグを符号化することを更に含み、第１の変換係数が、変換係数の第１のセットの１つであり、１より大のフラグが、第１の変換係数のレベルの絶対値が１より大であるか否かを示す、
条項２５に記載の符号化方法。
２７．
スキャンする第１のパスが、
１より大のフラグが、第１の変換係数のレベルの絶対値が１より大であると示すことに応じて、第１の変換係数の３より大のフラグを符号化することであって、３より大のフラグが、第１の変換係数のレベルの絶対値が３より大であるか否かを示し、スキャンする第１のパスを実行することが、残りのコンテキスト符号化されたビンの数がグループ限度未満である場合に停止される、ことと、
３より大のフラグが、第１の変換係数のレベルの絶対値が３より大であると示すことに応じて、第１の変換係数の５より大のフラグを符号化することであって、５より大のフラグが、第１の変換係数のレベルの絶対値が５より大であるか否かを示す、ことと、
５より大のフラグが、第１の変換係数のレベルの絶対値が５より大であると示すことに応じて、第１の変換係数の７より大のフラグを符号化することであって、７より大のフラグが、第１の変換係数のレベルの絶対値が７より大であるか否かを示す、ことと、
７より大のフラグが、第１の変換係数のレベルの絶対値が７より大であると示すことに応じて、第１の変換係数の９より大のフラグを符号化することであって、９より大のフラグが、第１の変換係数のレベルの絶対値が９より大であるか否かを示す、ことと
を更に含む、条項２６に記載の符号化方法。

２８．
スキャンする第２のパスが、
９より大のフラグが、第１の変換係数のレベルの絶対値が９より大であると示すことに応じて、第１の変換係数の残りの絶対レベルフラグを符号化することであって、残りの絶対レベルフラグが、第１の変換係数のレベルの残りの絶対値を示す、こと
を更に含む、条項２７に記載の符号化方法。
２９．
第１のパス及び第２のパスのそれぞれが、逆の順序でサブブロックの変換係数をスキャンすることによって実行される、条項２５～２８のいずれか一項に記載の符号化方法。
３０．
第１のパスを実行することの前に、サブブロックの変換係数を反転すること
を更に含む、条項２９に記載の符号化方法。
３１．
符号化方法が変換スキップ残差符号化方法である、条項２５～３０のいずれか一項に記載の符号化方法。
３２．
変換係数の第２のセットのそれぞれについてレベルの絶対値を２値化することが、
ゴロム・ライス符号を使用して絶対値を２値化すること
を更に含む、条項２５～３１のいずれか一項に記載の符号化方法。
３３．
ビデオデータのデコーダによって実施される復号方法であって、
ビデオフレームのサブブロックの変換係数をスキャンする第１のパスを実行することと、
サブブロックの変換係数をスキャンする第２のパスを実行することとを含み、
スキャンする第１のパスを実行することが、コンテキスト符号化されたビンの数が最大限度に到達すると停止され、
サブブロックの変換係数の第１のセットが第１のパスでスキャンされ、
スキャンする第１のパスが、変換係数の第１のセットのそれぞれについて、変換係数のレベルがゼロであるか否かを示す有意フラグを復号することを含み、
スキャンする第２のパスが、変換係数の第２のセットのそれぞれについてレベルの２値化された絶対値を復号することを含み、変換係数の第２のセットが第１のパスでスキャンされない、
復号方法。
３４．
スキャンする第１のパスが、
第１の変換係数の有意フラグが、第１の変換係数のレベルがゼロではないと示すことに応じて、第１の変換係数の１より大のフラグを復号することであって、第１の変換係数が、変換係数の第１のセットの１つであり、１より大のフラグが、第１の変換係数のレベルの絶対値が１より大であるか否かを示す、こと
を更に含む、条項３３に記載の復号方法。
３５．
スキャンする第１のパスが、
１より大のフラグが、第１の変換係数のレベルの絶対値が１より大であると示すことに応じて、第１の変換係数の３より大のフラグを復号することであって、３より大のフラグが、スキャンされている第１の変換係数のレベルの絶対値が３より大であるか否かを示し、スキャンする第１のパスを実行することが、残りのコンテキスト符号化されたビンの数がグループ限度未満である場合に停止される、ことと、
３より大のフラグが、第１の変換係数のレベルの絶対値が３より大であると示すことに応じて、第１の変換係数のそれぞれの５より大のフラグを復号することであって、５より大のフラグが、第１の変換係数のレベルの絶対値が５より大であるか否かを示す、ことと、

５より大のフラグが、第１の変換係数のレベルの絶対値が５より大であると示すことに応じて、第１の変換係数のそれぞれの７より大のフラグを復号することであって、７より大のフラグが、第１の変換係数のレベルの絶対値が７より大であるか否かを示す、ことと、

７より大のフラグが、第１の変換係数のレベルの絶対値が７より大であると示すことに応じて、第１の変換係数のそれぞれの９より大のフラグを復号することであって、９より大のフラグが、第１の変換係数のレベルの絶対値が９より大であるか否かを示す、ことと

を更に含む、条項３４に記載の復号方法。

３６．

スキャンする第２のパスが、

９より大のフラグが、スキャンされている第１の変換係数のレベルの絶対値が９より大であると示すことに応じて、第１の変換係数の残りの絶対レベルフラグを復号することであって、残りの絶対レベルフラグが、第１の変換係数のレベルの残りの絶対値を示す、こと

を更に含む、条項３５に記載の復号方法。

３７．

符号化方法が変換スキップ残差符号化方法である、条項３３～３６のいずれか一項に記載の復号方法。

３８．

変換係数の第２のセットのそれぞれのレベルの２値化された絶対値を復号することが、

ゴロム・ライス符号を使用して絶対値を復号すること

を更に含む、条項９～１３のいずれか一項に記載の復号方法。

３９．

ビデオデータを符号化するシステムであって、本システムが、

一組の命令を格納するメモリと、

プロセッサとを備え、プロセッサが、一組の命令を実行して、

ビデオフレームのサブブロックの変換係数をスキャンする第１のパスを実行することと、

サブブロックの変換係数をスキャンする第２のパスを実行することとをシステムに行わせるように構成され、

スキャンする第１のパスを実行することが、コンテキスト符号化されたビンの数が最大限度に到達すると停止され、

サブブロックの変換係数の第１のセットが第１のパスでスキャンされ、

スキャンする第１のパスが、変換係数の第１のセットのそれぞれについて、変換係数のレベルがゼロであるか否かを示す有意フラグを符号化することを含み、

スキャンする第２のパスが、変換係数の第２のセットのそれぞれのレベルの絶対値を２値化することを含み、変換係数の第２のセットが第１のパスでスキャンされない、

ビデオデータを符号化するシステム。

４０．

ビデオデータを復号するシステムであって、本システムが、

一組の命令を格納するメモリと、

プロセッサとを備え、プロセッサが、一組の命令を実行して、

ビデオフレームのサブブロックの変換係数をスキャンする第１のパスを実行することと、

サブブロックの変換係数をスキャンする第２のパスを実行することとをシステムに行わせるように構成され、

スキャンする第１のパスを実行することが、コンテキスト符号化されたビンの数が最大限度に到達すると停止され、

サブブロックの変換係数の第１のセットが第１のパスでスキャンされ、

スキャンする第１のパスが、変換係数の第１のセットのそれぞれについて、変換係数

のレベルがゼロであるか否かを示す有意フラグを復号することを含み、
スキャンする第２のパスが、変換係数の第２のセットのそれぞれのレベルの２値化された絶対値を復号することを含み、変換係数の第２のセットが第１のパスでスキャンされない、
ビデオデータを復号するシステム。
４１．
ビデオフレームのルーマ成分のコンテキスト変数の第１のセットを生成することと、
ビデオフレームのクロマ成分のコンテキスト変数の第２のセットを生成することと、
ビデオフレームのサブブロックを生成することと、
コンテキスト変数の第１のセットとコンテキスト変数の第２のセットとにしたがってサブブロックの変換係数の第１のセットを符号化することと
を含む、ビデオデータのエンコーダによって実施される符号化方法。
４２．
変換係数の第１のセットを符号化することが、
コンテキスト変数の第１のセットからの３つのコンテキスト変数とコンテキスト変数の第２のセットからの３つのコンテキスト変数とにしたがって、変換係数の第１のセットの変換係数の有意フラグを符号化することを含み、有意フラグが、変換係数のレベルがゼロであるか否かを示す、
条項４１に記載の符号化方法。
４３．
コンテキスト変数の第１のセットからの３つのコンテキスト変数と、コンテキスト変数の第２のセットからの３つのコンテキスト変数とを生成することを更に含み、生成することが、色成分インデックスと、ビデオフレームに対するサブブロックのロケーションを指定するルーマロケーションと、対象の係数スキャンロケーションとに基づく、
条項４２に記載の符号化方法。
４４．
変換係数の第１のセットを符号化することが、
コンテキスト変数の第１のセットからの４つのコンテキスト変数とコンテキスト変数の第２のセットからの４つのコンテキスト変数とにしたがって、変換係数の第１のセットの変換係数のｘより大のフラグを符号化することを含み、ｘより大のフラグが、変換係数のレベルの絶対値がｘより大であるか否かを示す、
条項４１～４３のいずれか一項に記載の符号化方法。
４５．
コンテキスト変数の第１のセットからの４つのコンテキスト変数と、コンテキスト変数の第２のセットからの４つのコンテキスト変数とを生成することを更に含み、生成することが、色成分インデックスと、ビデオフレームに対するサブブロックのロケーションを指定するルーマロケーションと、対象の係数スキャンロケーションとに基づく、
条項４４に記載の符号化方法。
４６．
変換係数の第１のセットを符号化することが、
コンテキスト変数の第１のセットからの１つのコンテキスト変数とコンテキスト変数の第２のセットからの１つのコンテキスト変数とにしたがって、変換係数の第１のセットの変換係数のパリティフラグを符号化することを含み、パリティフラグが、変換係数のレベルの絶対値のパリティであることを示す、
条項４１～４５のいずれか一項に記載の符号化方法。
４７．
色成分インデックスにしたがって、コンテキスト変数の第１のセットからの１つのコンテキスト変数と、コンテキスト変数の第２のセットからの１つのコンテキスト変数とを生成すること
を更に含む、条項４６に記載の符号化方法。

４８．
変換係数の第１のセットを符号化することが、
コンテキスト変数の第１のセットからの６つのコンテキスト変数とコンテキスト変数の第２のセットからの６つのコンテキスト変数とにしたがって、変換係数の第１のセットの変換係数の係数符号フラグを符号化することを含み、係数符号フラグが、変換係数の値の符号を示す、
条項４１～４７のいずれか一項に記載の符号化方法。
４９．
コンテキスト変数の第１のセットからの６つのコンテキスト変数と、コンテキスト変数の第２のセットからの６つのコンテキスト変数とを生成することを更に含み、生成することが、ビデオフレームに隣接するビデオフレームの有意係数の数と、ビデオフレームに対するサブブロックのロケーションを指定するルーマロケーションと、対象の係数スキャンロケーションとに基づく、
条項４８に記載の方法。
５０．
変換係数の第１のセットを符号化することが、
コンテキスト変数の第１のセットからの３つのコンテキスト変数とコンテキスト変数の第２のセットからの３つのコンテキスト変数とにしたがって、変換係数の第１のセットの変換係数のサブブロックフラグを符号化すること
を含む、条項４１～４９のいずれか一項に記載の方法。
５１．
コンテキスト変数の第１のセットからの３つのコンテキスト変数と、コンテキスト変数の第２のセットからの３つのコンテキスト変数とを生成することを更に含み、生成することが、ビデオフレームに隣接するビデオフレームの有意係数の数と、ビデオフレームに対するサブブロックのロケーションを指定するルーマロケーションと、対象の係数スキャンロケーションとに基づく、
条項５０に記載の方法。
５２．
ビデオビットストリームを受信することと、
ビデオビットストリームを複数のサブブロックに分割することと、
サブブロックのルーマ成分のコンテキスト変数の第１のセットを生成することと、
サブブロックのクロマ成分のコンテキスト変数の第２のセットを生成することと、
コンテキスト変数の第１のセットとコンテキスト変数の第２のセットとにしたがってサブブロックの変換係数の第１のセットをコンテキスト符号化することと
を含む、ビデオ処理方法。
５３．
ビデオフレームを受信することと、
ビデオフレームを複数のサブブロックに分割することと、
ビデオフレームのルーマ成分のコンテキスト変数の第１のセットを生成することと、
ビデオフレームのクロマ成分のコンテキスト変数の第２のセットを生成することと、
コンテキスト変数の第１のセットとコンテキスト変数の第２のセットとにしたがってサブブロックの変換係数の第１のセットを復号することと
を含む、ビデオデータのデコーダによって実施される復号方法。
５４．
変換係数の第１のセットを復号することが、
コンテキスト変数の第１のセットからの３つのコンテキスト変数とコンテキスト変数の第２のセットからの３つのコンテキスト変数とにしたがって、変換係数の第１のセットの変換係数の有意フラグを復号することを含み、有意フラグが、変換係数のレベルがゼロであるか否かを示す、
条項５３に記載の復号方法。

５５．
コンテキスト変数の第１のセットからの３つのコンテキスト変数と、コンテキスト変数の第２のセットからの３つのコンテキスト変数とを生成することを更に含み、生成することが、色成分インデックスと、ビデオフレームの左上サンプルに対する対象の変換ブロックの左上サンプルを指定するルーマロケーションと、対象の係数スキャンロケーションとに基づく、
条項５４に記載の復号方法。
５６．
変換係数の第１のセットを復号することが、
コンテキスト変数の第１のセットからの４つのコンテキスト変数とコンテキスト変数の第２のセットからの４つのコンテキスト変数とにしたがって、変換係数の第１のセットの変換係数のｘより大のフラグを復号すること
を含む、条項５３～５５のいずれか一項に記載の復号方法。
５７．
コンテキスト変数の第１のセットからの４つのコンテキスト変数と、コンテキスト変数の第２のセットからの４つのコンテキスト変数とを生成することを更に含み、生成することが、色成分インデックスと、ビデオフレームの左上サンプルに対する対象の変換ブロックの左上サンプルを指定するルーマロケーションと、対象の係数スキャンロケーションとに基づく、
条項５６に記載の復号方法。
５８．
変換係数の第１のセットを復号することが、
コンテキスト変数の第１のセットからの１つのコンテキスト変数とコンテキスト変数の第２のセットからの１つのコンテキスト変数とにしたがって、変換係数の第１のセットの変換係数のパリティフラグを復号すること
を含む、条項５３～５７のいずれか一項に記載の復号方法。
５９．
コンテキスト変数の第１のセットからの１つのコンテキスト変数と、コンテキスト変数の第２のセットからの１つのコンテキスト変数とを生成することを更に含み、生成することが、色成分インデックスに基づく、
条項５８に記載の復号方法。
６０．
変換係数の第１のセットを復号することが、
コンテキスト変数の第１のセットからの６つのコンテキスト変数とコンテキスト変数の第２のセットからの６つのコンテキスト変数とにしたがって、変換係数の第１のセットの変換係数のコンテキスト係数符号フラグを復号すること
を含む、条項５３～５９のいずれか一項に記載の復号方法。
６１．
コンテキスト変数の第１のセットからの６つのコンテキスト変数と、コンテキスト変数の第２のセットからの６つのコンテキスト変数とを生成することを更に含み、生成することが、ビデオフレームの上隣と左隣とのビデオフレームの有意係数の数と、ビデオフレームの左上サンプルに対する対象の変換ブロックの左上サンプルを指定するルーマロケーションと、対象の係数スキャンロケーションとに基づく、
条項６０に記載の復号方法。
６２．
変換係数の第１のセットを復号することが、
コンテキスト変数の第１のセットからの３つのコンテキスト変数とコンテキスト変数の第２のセットからの３つのコンテキスト変数とにしたがって、変換係数の第１のセットの変換係数のサブブロックフラグを復号すること
を含む、条項５３～６１のいずれか一項に記載の復号方法。

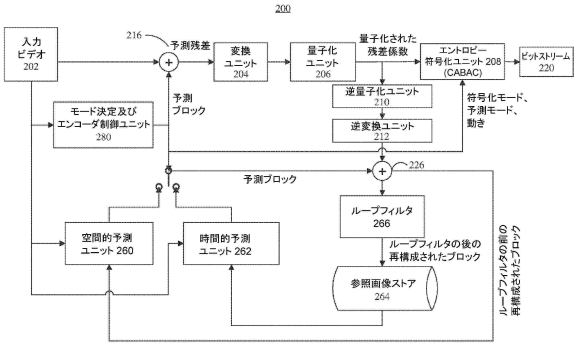
６３．
コンテキスト変数の第１のセットからの３つのコンテキスト変数と、コンテキスト変数の第２のセットからの３つのコンテキスト変数とを生成することを更に含み、生成することが、ビデオフレームの上隣と左隣とのビデオフレームの有意係数の数と、ビデオフレームの左上サンプルに対する対象変換ブロックの左上サンプルを指定するルーマロケーションと、対象の係数スキャンロケーションとに基づく、
条項６２に記載の復号方法。
６４．
ビデオデータを符号化するシステムであって、本システムが、
一組の命令を格納するメモリと、
プロセッサとを備え、プロセッサが、一組の命令を実行して、
ビデオフレームのルーマ成分のコンテキスト変数の第１のセットを生成することと、
ビデオフレームのクロマ成分のコンテキスト変数の第２のセットを生成することと、
ビデオフレームのサブブロックを生成することと、
コンテキスト変数の第１のセットとコンテキスト変数の第２のセットとにしたがってサブブロックの変換係数の第１のセットを符号化することと
をシステムに行わせるように構成される、
ビデオデータを符号化するシステム。
６５．
ビデオデータを復号するシステムであって、本システムが、
一組の命令を格納するメモリと、
プロセッサとを備え、プロセッサが、一組の命令を実行して、
ビデオフレームを受信することと、
ビデオフレームを複数のサブブロックに分割することと、
ビデオフレームのルーマ成分のコンテキスト変数の第１のセットを生成することと、
ビデオフレームのクロマ成分のコンテキスト変数の第２のセットを生成することと、
コンテキスト変数の第１のセットとコンテキスト変数の第２のセットとにしたがってサブブロックの変換係数の第１のセットを復号することと
をシステムに行わせるように構成される、
ビデオデータを復号するシステム。
６６．
一組の命令を格納する非一時的コンピュータ可読媒体であって、一組の命令は、ビデオデータを符号化するための方法を装置に開始させるように、装置の１つ又は複数のプロセッサによって実行可能であり、本方法が、
ビデオフレームのサブブロックの変換係数をスキャンする第１のパスを実行することを含み、
スキャンする第１のパスが、変換係数のパリティレベルフラグをバイパス符号化することを含み、パリティレベルフラグが変換係数のレベルの絶対値のパリティを示す、
非一時的コンピュータ可読媒体。
６７．
一組の命令を格納する非一時的コンピュータ可読媒体であって、一組の命令が、ビデオデータを復号するための方法を装置に開始させるように、装置の１つ又は複数のプロセッサによって実行可能であり、本方法が、
ビデオフレームのサブブロックの変換係数をスキャンする第１のパスを実行することを含み、
スキャンする第１のパスが、変換係数のパリティレベルフラグをバイパス復号することを含み、パリティレベルフラグが変換係数のレベルの絶対値のパリティを示す、
非一時的コンピュータ可読媒体。
６８．
一組の命令を格納する非一時的コンピュータ可読媒体であって、一組の命令が、ビデオ

データを符号化するための方法を装置に開始させるように、装置の１つ又は複数のプロセッサによって実行可能であり、本方法が、
ビデオフレームのサブブロックの変換係数をスキャンする第１のパスを実行することと、
サブブロックの変換係数をスキャンする第２のパスを実行することとを含み、
スキャンする第１のパスを実行することが、コンテキスト符号化されたビンの数が最大限度に到達すると停止され、
サブブロックの変換係数の第１のセットが第１のパスでスキャンされ、
スキャンする第１のパスが、変換係数の第１のセットのそれぞれについて、変換係数のレベルがゼロであるか否かを示す有意フラグを符号化することを含み、
スキャンする第２のパスが、変換係数の第２のセットのそれぞれのレベルの絶対値を２値化することを含み、変換係数の第２のセットが第１のパスでスキャンされない、
非一時的コンピュータ可読媒体。
６９．
一組の命令を格納する非一時的コンピュータ可読媒体であって、一組の命令が、ビデオデータを復号するための方法を装置に開始させるように、装置の１つ又は複数のプロセッサによって実行可能であり、本方法が、
ビデオフレームのサブブロックの変換係数をスキャンする第１のパスを実行することと、
サブブロックの変換係数をスキャンする第２のパスを実行することとを含み、
スキャンする第１のパスを実行することが、コンテキスト符号化されたビンの数が最大限度に到達すると停止され、
サブブロックの変換係数の第１のセットが第１のパスでスキャンされ、
スキャンする第１のパスが、変換係数の第１のセットのそれぞれについて、変換係数のレベルがゼロであるか否かを示す有意フラグを復号することを含み、
スキャンする第２のパスが、変換係数の第２のセットのそれぞれのレベルの２値化された絶対値を復号することを含み、変換係数の第２のセットが第１のパスでスキャンされない、
非一時的コンピュータ可読媒体。
７０．
一組の命令を格納する非一時的コンピュータ可読媒体であって、一組の命令が、ビデオデータを符号化するための方法を装置に開始させるように、装置の１つ又は複数のプロセッサによって実行可能であり、本方法が、
ビデオフレームのルーマ成分のコンテキスト変数の第１のセットを生成することと、
ビデオフレームのクロマ成分のコンテキスト変数の第２のセットを生成することと、
ビデオフレームのサブブロックを生成することと、
コンテキスト変数の第１のセットとコンテキスト変数の第２のセットとにしたがってサブブロックの変換係数の第１のセットを符号化することと
を含む、非一時的コンピュータ可読媒体。
７１．
一組の命令を格納する非一時的コンピュータ可読媒体であって、一組の命令が、ビデオデータを復号するための方法を装置に開始させるように、装置の１つ又は複数のプロセッサによって実行可能であり、本方法が、
ビデオフレームを受信することと、
ビデオフレームを複数のサブブロックに分割することと、
ビデオフレームのルーマ成分のコンテキスト変数の第１のセットを生成することと、
ビデオフレームのクロマ成分のコンテキスト変数の第２のセットを生成することと、
コンテキスト変数の第１のセットとコンテキスト変数の第２のセットとにしたがってサブブロックの変換係数の第１のセットを復号することと
を含む、非一時的コンピュータ可読媒体。
【０１１７】

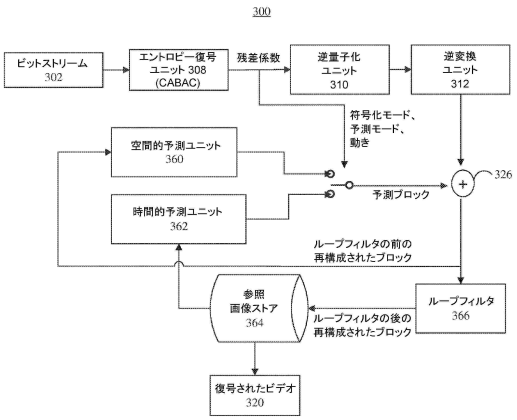
[00135] 図面及び明細書において、例示的な実施形態を開示してきた。しかしながら、これらの実施形態には多くの変形及び修正を加えることができる。したがって、特定の用語が使用されているが、それらは一般的且つ説明的な意味でのみ使用されており、限定の目的では使用されていない。

【図面】

【図１】




【図２】




【図３－１】

| residual_coding( x0, y0, log2TbWidth, log2TbHeight, cIdx ) { | Descriptor |
|---|---|
| if( ( tu_mts_idx[ x0 ][ y0 ] > 0 \|\| ( cu_sbt_flag && log2TbWidth < 6 && log2TbHeight < 6 ) ) && cIdx = = 0 && log2TbWidth > 4 ) | |
| log2ZoTbWidth = 4 | |
| Else | |
| log2ZoTbWidth = Min( log2TbWidth, 5 ) | |
| MaxCcbs = 2 * ( 1 << log2TbWidth ) * ( 1<< log2TbHeight ) | |
| if( tu_mts_idx[ x0 ][ y0 ] > 0 \|\| ( cu_sbt_flag && log2TbWidth < 6 && log2TbHeight < 6 ) ) && cIdx = = 0 && log2TbHeight > 4 ) | |
| 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 | |
| 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 ) | |
| lastScanPos = numSbCoeff | |
| lastSubBlock = ( 1 << ( log2TbWidth + log2TbHeight – ( log2SbW + log2SbH ) ) ) – 1 | |
| do { | |

FIG. 3

【図３－２】

| | |
|---|---|
| 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 ] | |
| } while( ( xC != LastSignificantCoeffX ) \|\| ( yC != LastSignificantCoeffY ) ) | |
| QState = 0 | |
| for( i = lastSubBlock; i >= 0; i– – ) { | |
| startQStateSb = QState | |
| xS = DiagScanOrder[ log2TbWidth – log2SbW ][ log2TbHeight – log2SbH ][ i ][ 0 ] | |
| yS = DiagScanOrder[ log2TbWidth – log2SbW ][ log2TbHeight – log2SbH ][ i ][ 1 ] | |
| inferSbDcSigCoeffFlag = 0 | |
| if( ( i < lastSubBlock ) && ( i > 0 ) ) { | |
| **coded_sub_block_flag**[ xS ][ yS ] | ae(v) |
| inferSbDcSigCoeffFlag = 1 | |
| } | |
| firstSigScanPosSb = numSbCoeff | |
| lastSigScanPosSb = –1 | |
| remBinsPass1 = ( ( log2SbW + log2SbH ) < 4 ? 8 : 32 ) | |
| firstPosMode0 = ( i == lastSubBlock ? lastScanPos : numSbCoeff – 1 ) | |
| firstPosMode1 = –1 | |
| 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( coded_sub_block_flag[ xS ][ yS ] && ( n > 0 \|\| !inferSbDcSigCoeffFlag ) && ( xC != LastSignificantCoeffX \|\| yC != LastSignificantCoeffY ) ) { | |
| **sig_coeff_flag**[ xC ][ yC ] | ae(v) |
| remBinsPass1– – | |
| if( sig_coeff_flag[ xC ][ yC ] ) | |
| inferSbDcSigCoeffFlag = 0 | |
| } | |
| if( sig_coeff_flag[ xC ][ yC ] ) { | |
| if( !transform_skip_flag[ x0 ][ y0 ] ) { | |
| numSigCoeff++ | |

FIG. 3 (continued)

【図３－３】

```
            if( ( n >= 8 && i == 0 && ( log2TbWidth == 2 || log2TbWidth == 3 )
              && ( log2TbWidth == log2TbHeight ) ) || ( ( i == 1 || i == 2 )
              && log2TbWidth >= 3 && log2TbHeight >= 3 ) )
              numZeroOutSigCoeff++
          }
          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--
          }
          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 ]
        if( dep_quant_enabled_flag )
          QState = QStateTransTable[ QState ][ AbsLevelPass1[ xC ][ yC ] & 1 ]
        if( remBinsPass1 < 4 )
          firstPosMode1 = n - 1
      }
      for( n = numSbCoeff - 1; n >= firstPosMode1; n-- ) {
        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 ]
      }
      for( n = firstPosMode1; n>= 0; n-- ) {
        xC = ( xS << log2SbW ) + DiagScanOrder[ log2SbW ][ log2SbH ][ n ][ 0 ]
        yC = ( yS << log2SbH ) + DiagScanOrder[ log2SbW ][ log2SbH ][ n ][ 1 ]
        dec_abs_level[ n ]                                                ae(v)
        if(AbsLevel[ xC ][ yC ] > 0 )
          firstSigScanPosSb = n
        if( dep_quant_enabled_flag )
          QState = QStateTransTable[ QState ][ AbsLevel[ xC ][ yC ] & 1 ]
      }
      if( dep_quant_enabled_flag || !sign_data_hiding_enabled_flag )
        signHidden = 0
      Else
```

FIG. 3 (continued)

【図３－４】

```
        signHidden = ( 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 ) &&
          ( !signHidden || ( n != firstSigScanPosSb ) ) )
          coeff_sign_flag[ n ]                                            ae(v)
      }
      if( dep_quant_enabled_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 ]
          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 ][ par_level_flag[ n ] ]
        } 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( signHidden ) {
                sumAbsLevel += AbsLevel[ xC ][ yC ]
                if( ( n == firstSigScanPosSb ) && ( sumAbsLevel % 2 ) == 1 ) )
                  TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ] =
                    -TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ]
              }
            }
          }
        }
      }
```

FIG. 3 (continued)

【図４－１】

| residual_ts_coding( x0, y0, log2TbWidth, log2TbHeight, cIdx ) { | Descriptor |
|---|---|
| log2SbSize = ( Min( log2TbWidth, log2TbHeight ) < 2 ? 1 : 2 ) | |
| numSbCoeff = 1 << ( log2SbSize << 1 ) | |
| lastSubBlock = ( 1 << ( log2TbWidth + log2TbHeight - 2 * log2SbSize ) ) - 1 | |
| inferSbCbf = 1 | |
| MaxCcbs = 2 * ( 1 << log2TbWidth ) * ( 1<< log2TbHeight ) | |
| for( i =0; i <= lastSubBlock; i++ ) { | |
| xS = DiagScanOrder[ log2TbWidth - log2SbSize ][ log2TbHeight - log2SbSize ][ i ][ 0 ] | |
| yS = DiagScanOrder[ log2TbWidth - log2SbSize ][ log2TbHeight - log2SbSize ][ i ][ 1 ] | |
| if( ( i != lastSubBlock || !inferSbCbf ) { | |
| coded_sub_block_flag[ xS ][ yS ] | ae(v) |
| MaxCcbs-- | |
| } | |
| if( coded_sub_block_flag[ xS ][ yS ] && i < lastSubBlock ) | |
| inferSbCbf = 0 | |
| /* First scan pass */ | |
| inferSbSigCoeffFlag = 1 | |
| for( n = 0; n <= numSbCoeff - 1; n++ ) { | |
| xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ] | |
| yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ] | |
| if( coded_sub_block_flag[ xS ][ yS ] && | |
| ( n != numSbCoeff - 1 || !inferSbSigCoeffFlag ) ) { | |
| sig_coeff_flag[ xC ][ yC ] | ae(v) |
| MaxCcbs-- | |
| if( sig_coeff_flag[ xC ][ yC ] ) | |
| inferSbSigCoeffFlag = 0 | |
| } | |
| if( sig_coeff_flag[ xC ][ yC ] ) { | |
| coeff_sign_flag[ n ] | ae(v) |
| MaxCcbs-- | |
| abs_level_gtx_flag[ n ][ 0 ] | ae(v) |
| MaxCcbs-- | |
| if( abs_level_gtx_flag[ n ][ 0 ] ) { | |
| par_level_flag[ n ] | ae(v) |
| MaxCcbs-- | |
| } | |
| } | |
| AbsLevelPassX[ xC ][ yC ] = sig_coeff_flag[ xC ][ yC ] + par_level_flag[ n ] + abs_level_gtx_flag[ n ][ 0 ] | |

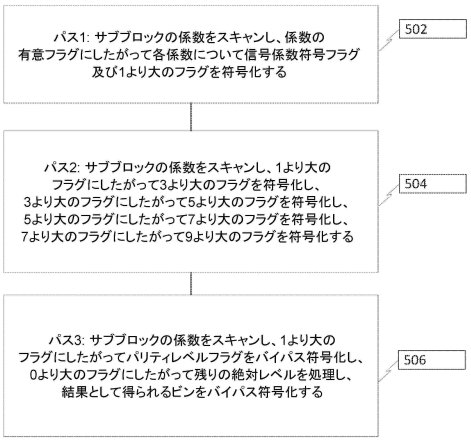
FIG. 4

【図４－２】

```
    }
  /* Greater than X scan passes (numGtXFlags=5) */
    for( j = 1; j < 5; j++ ) {
      for( n = 0; n <= numSbCoeff - 1; n++ ) {
        xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ]
        yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ]
        if( abs_level_gtx_flag[ n ][ j - 1 ] )
          abs_level_gtx_flag[ n ][ j ]                                    ae(v)
        MaxCcbs--
        AbsLevelPassX[ xC ][ yC ] += 2 * abs_level_gtx_flag[ n ][ j ]
      }
    }
  /* remainder scan pass */
    for( n = 0; n <= numSbCoeff - 1; n++ ) {
      xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ]
      yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ]
      if( abs_level_gtx_flag[ n ][ 4 ] )
        abs_remainder[ n ]                                                ae(v)
      TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ] = ( 1 - 2 * coeff_sign_flag[ n ] ) *
                                                        ( AbsLevelPassX[ xC ][ yC ] + abs_remainder[ n ] )
    }
  }
}
```

FIG. 4 (continued)

【図５】




【図６－１】

| residual_ts_coding( x0, y0, log2TbWidth, log2TbHeight, cIdx ) { | Descriptor |
|---|---|
| log2SbSize = ( Min( log2TbWidth, log2TbHeight ) < 2 ? 1 : 2 ) | |
| numSbCoeff = 1 << ( log2SbSize << 1 ) | |
| lastSubBlock = ( 1 << ( log2TbWidth + log2TbHeight – 2 * log2SbSize ) ) – 1 | |
| inferSbCbf = 1 | |
| MaxCcbs = 2 * ( 1 << log2TbWidth ) * ( 1<< log2TbHeight ) | |
| for( i =0; i <= lastSubBlock; i++ ) { | |
| xS = DiagScanOrder[ log2TbWidth – log2SbSize ][ log2TbHeight – log2SbSize ][ i ][ 0 ] | |
| yS = DiagScanOrder[ log2TbWidth – log2SbSize ][ log2TbHeight – log2SbSize ][ i ][ 1 ] | |
| if( ( i != lastSubBlock ) || !inferSbCbf ) { | |
| **coded_sub_block_flag**[ xS ][ yS ] | ae(v) |
| MaxCcbs– – | |
| } | |
| if( coded_sub_block_flag[ xS ][ yS ] && i < lastSubBlock ) | |
| inferSbCbf = 0 | |
| /* First scan pass */ | |
| inferSbSigCoeffFlag = 1 | |
| for( n = 0; n <= numSbCoeff – 1; n++ ) { | |
| xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ] | |
| yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ] | |
| if( coded_sub_block_flag[ xS ][ yS ] && | |
| ( n != numSbCoeff – 1 || !inferSbSigCoeffFlag ) ) { | |
| **sig_coeff_flag**[ xC ][ yC ] | ae(v) |
| MaxCcbs– – | |
| if( sig_coeff_flag[ xC ][ yC ] ) | |
| inferSbSigCoeffFlag = 0 | |
| } | |
| if( sig_coeff_flag[ xC ][ yC ] ) { | |
| **coeff_sign_flag**[ n ] | ae(v) |
| MaxCcbs– – | |
| **abs_level_gtx_flag**[ n ][ 0 ] | ae(v) |
| MaxCcbs– – | |
| } | |
| AbsLevelPassX[ xC ][ yC ] = | |
| sig_coeff_flag[ xC ][ yC ] + abs_level_gtx_flag[ n ][ 0 ] | |
| } | |
| /* Greater than X scan passes (numGtXFlags=5) */ | |
| for( n = 0; n <= numSbCoeff – 1; n++ ) { | |

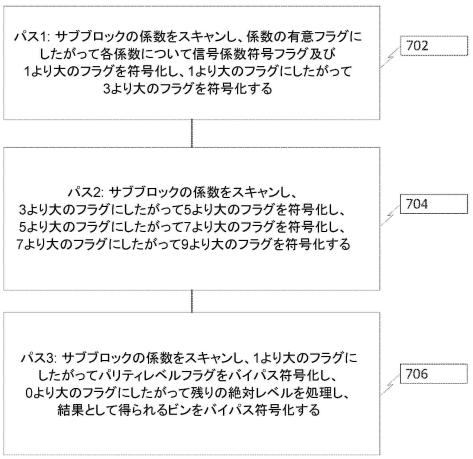
FIG. 6

【図６－２】

| | |
|---|---|
| xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ] | |
| yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ] | |
| *for( j = 1; j < 5; j++ ) {* | |
| *if( abs_level_gtx_flag[ n ][ j – 1 ] )* | |
| ***abs_level_gtx_flag[ n ][ j ]*** | *ae(v)* |
| *MaxCcbs– –* | |
| *AbsLevelPassX[ xC ][ yC ] += 2 * abs_level_gtx_flag[ n ][ j ]* | |
| *}* | |
| } | |
| /* remainder scan pass */ | |
| for( n = 0; n <= numSbCoeff – 1; n++ ) { | |
| xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ] | |
| yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ] | |
| *if( abs_level_gtx_flag[ n ][ 0 ] )* | |
| ***par_level_flag[ n ]*** | *ae(v)* |
| if( abs_level_gtx_flag[ n ][ 4 ] ) | |
| **abs_remainder**[ n ] | ae(v) |
| TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ] = ( 1 – 2 * coeff_sign_flag[ n ] ) * ( AbsLevelPassX[ xC ][ yC ] + par_level_flag[ n ] + abs_remainder[ n ] ) | |
| } | |
| } | |
| } | |

FIG. 6 (continued)

【図７】



【図８－１】

| residual_ts_coding( x0, y0, log2TbWidth, log2TbHeight, cIdx ) { | Descriptor |
|---|---|
| log2SbSize = ( Min( log2TbWidth, log2TbHeight ) < 2 ? 1 : 2 ) | |
| numSbCoeff = 1 << ( log2SbSize << 1 ) | |
| lastSubBlock = ( 1 << ( log2TbWidth + log2TbHeight – 2 * log2SbSize ) ) – 1 | |
| inferSbCbf = 1 | |
| MaxCcbs = 2 * ( 1 << log2TbWidth ) * ( 1<< log2TbHeight ) | |
| for( i =0; i <= lastSubBlock; i++ ) { | |
| xS = DiagScanOrder[ log2TbWidth – log2SbSize ][ log2TbHeight – log2SbSize ][ i ][ 0 ] | |
| yS = DiagScanOrder[ log2TbWidth – log2SbSize ][ log2TbHeight – log2SbSize ][ i ][ 1 ] | |
| if( ( i != lastSubBlock && !inferSbCbf ) { | |
| **coded_sub_block_flag**[ xS ][ yS ] | ae(v) |
| MaxCcbs– – | |
| } | |
| if( coded_sub_block_flag[ xS ][ yS ] && i < lastSubBlock ) | |
| inferSbCbf = 0 | |
| /* First scan pass */ | |
| inferSbSigCoeffFlag = 1 | |
| for( n = 0; n <= numSbCoeff – 1; n++ ) { | |
| xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ] | |
| yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ] | |
| if( coded_sub_block_flag[ xS ][ yS ] && ( n != numSbCoeff – 1 || !inferSbSigCoeffFlag ) ) { | |
| **sig_coeff_flag**[ xC ][ yC ] | ae(v) |
| MaxCcbs– – | |
| if( sig_coeff_flag[ xC ][ yC ] ) | |
| inferSbSigCoeffFlag = 0 | |
| } | |
| if( sig_coeff_flag[ xC ][ yC ] ) { | |
| **coeff_sign_flag**[ n ] | ae(v) |
| MaxCcbs– – | |
| **abs_level_gtx_flag**[ n ][ 0 ] | ae(v) |
| MaxCcbs– – | |
| *if( abs_level_gtx_flag[ n ][ 0 ] )* | |
| ***abs_level_gtx_flag****[ n ][ 1 ]* | *ae(v)* |
| *MaxCcbs– –* | |
| } | |
| } | |

FIG. 8

【図８－２】

| | |
|---|---|
| AbsLevelPassX[ xC ][ yC ] = sig_coeff_flag[ xC ][ yC ] + abs_level_gtx_flag[ n ][ 0 ] + 2 * abs_level_gtx_flag[ n ][ 1 ] | |
| } | |
| /* Greater than X scan passes (numGtXFlags=5) */ | |
| for( n = 0; n <= numSbCoeff – 1; n++ ) { | |
| xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ] | |
| yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ] | |
| *for( j = 2; j < 5; j++ ) {* | |
| *if( abs_level_gtx_flag[ n ][ j – 1 ] )* | |
| ***abs_level_gtx_flag****[ n ][ j ]* | *ae(v)* |
| *MaxCcbs– –* | |
| *AbsLevelPassX[ xC ][ yC ] += 2 * abs_level_gtx_flag[ n ][ j ]* | |
| *}* | |
| } | |
| /* remainder scan pass */ | |
| for( n = 0; n <= numSbCoeff – 1; n++ ) { | |
| xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ] | |
| yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ] | |
| *if( abs_level_gtx_flag[ n ][ 0 ] )* | |
| ***par_level_flag****[ n ]* | *ae(v)* |
| if( abs_level_gtx_flag[ n ][ 4 ] ) | |
| **abs_remainder**[ n ] | ae(v) |
| TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ] = ( 1 – 2 * coeff_sign_flag[ n ] ) * ( AbsLevelPassX[ xC ][ yC ] + par_level_flag[ n ] + abs_remainder[ n ] ) | |
| } | |
| } | |
| } | |

FIG. 8 (continued)

【図９】

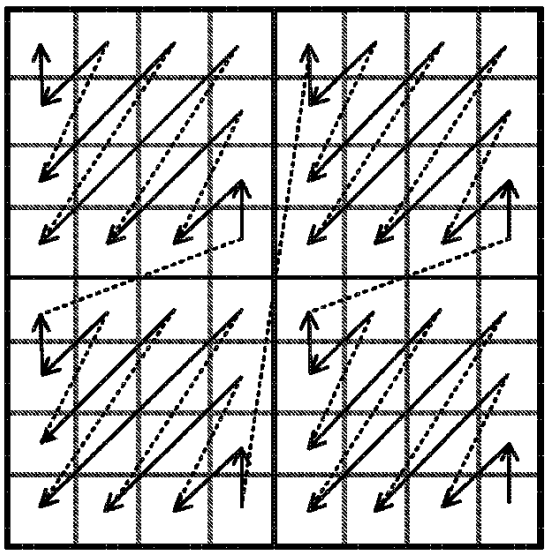


FIG. 9

【図１０Ａ】

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
| 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |
| 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |

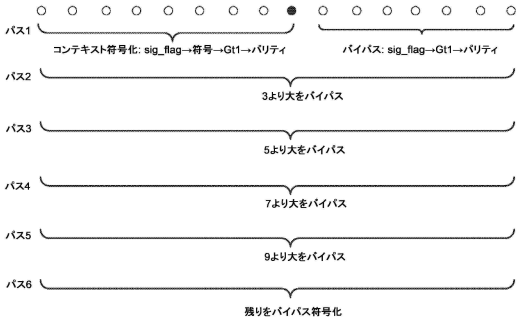
FIG. 10A

【図１０Ｂ】

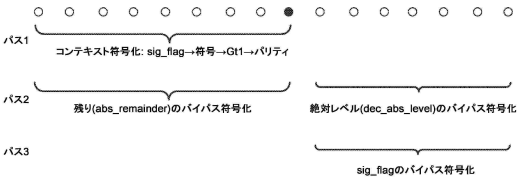
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 |
| 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 |
| 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 |
| 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

FIG. 10B

【図１１】




【図１２】

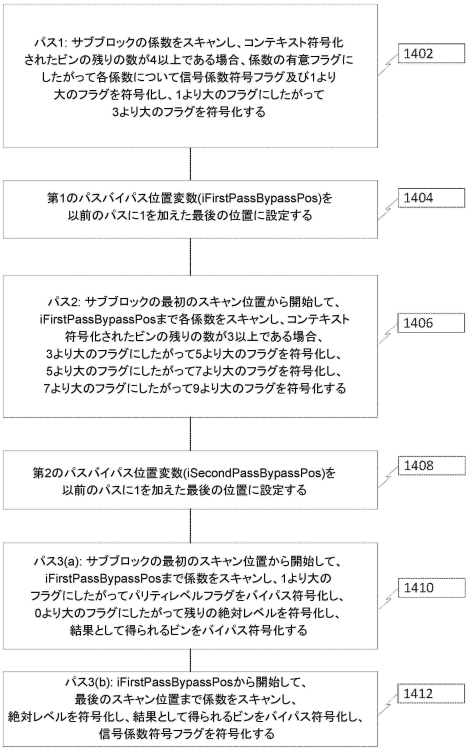



【図１３】

| 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 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| locSumAbs | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| cRiceParam | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |

FIG. 13

【図１４】




【図１５－１】

| residual_ts_coding( x0, y0, log2TbWidth, log2TbHeight, cIdx ) { | Descriptor |
|---|---|
| log2SbSize = ( Min( log2TbWidth, log2TbHeight ) < 2 ? 1 : 2 ) | |
| numSbCoeff = 1 << ( log2SbSize << 1 ) | |
| lastSubBlock = ( 1 << ( log2TbWidth + log2TbHeight – 2 * log2SbSize ) ) – 1 | |
| inferSbCbf = 1 | |
| MaxCcbs = 2 * ( 1 << log2TbWidth ) * ( 1<< log2TbHeight ) | |
| for( i =0; i <= lastSubBlock; i++ ) { | |
| xS = DiagScanOrder[ log2TbWidth – log2SbSize ][ log2TbHeight – log2SbSize ][ i ][ 0 ] | |
| yS = DiagScanOrder[ log2TbWidth – log2SbSize ][ log2TbHeight – log2SbSize ][ i ][ 1 ] | |
| if( ( i != lastSubBlock ) \|\| !inferSbCbf ) { | |
| **coded_sub_block_flag**[ xS ][ yS ] | ae(v) |
| MaxCcbs– – | |
| } | |
| if( coded_sub_block_flag[ xS ][ yS ] && i < lastSubBlock ) | |
| inferSbCbf = 0 | |
| /* First scan pass */ | |
| inferSbSigCoeffFlag = 1 | |
| for( n = 0; n <= numSbCoeff – 1 *&& MaxCcbs >= 4*; n++ ) { | |
| xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ] | |
| yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ] | |
| if( coded_sub_block_flag[ xS ][ yS ] && ( n != numSbCoeff – 1 \|\| !inferSbSigCoeffFlag ) ) { | |
| **sig_coeff_flag**[ xC ][ yC ] | ae(v) |
| MaxCcbs– – | |
| if( sig_coeff_flag[ xC ][ yC ] ) | |
| inferSbSigCoeffFlag = 0 | |
| } | |
| if( sig_coeff_flag[ xC ][ yC ] ) { | |
| **coeff_sign_flag**[ n ] | ae(v) |
| MaxCcbs– – | |
| **abs_level_gtx_flag**[ n ][ 0 ] | ae(v) |
| MaxCcbs– – | |
| *if( abs_level_gtx_flag[ n ][ 0 ] )*{ | |
| ***abs_level_gtx_flag****[ n ][ 1 ]* | *ae(v)* |
| *MaxCcbs– –* | |
| } | |
| } | |

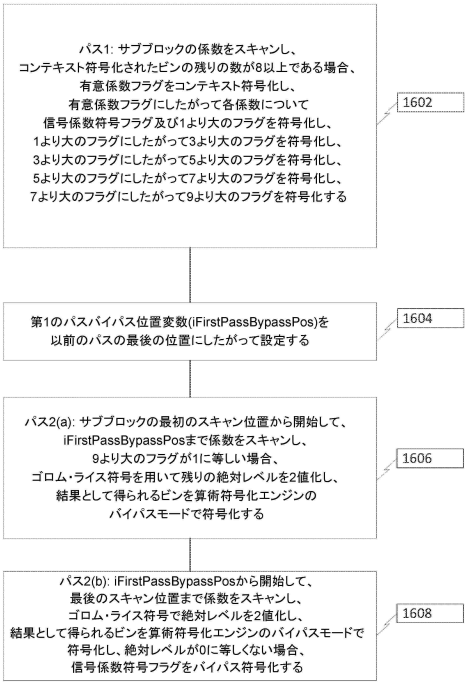
**FIG. 15**

【図１５－２】

| | |
|---|---|
| AbsLevelPassX[ xC ][ yC ] = sig_coeff_flag[ xC ][ yC ] + abs_level_gtx_flag[ n ][ 0 ] + 2 * abs_level_gtx_flag[ n ][ 1 ] | |
| } | |
| iFirstPassBypassPos = n; | |
| /* second scan pass*/ | |
| *for( n = 0; n < iFirstPassBypassPos && MaxCcbs >= 3; n++ ) {* | |
| xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ] | |
| yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ] | |
| for( j = 2; j < 5; j++ ) { | |
| *if( abs_level_gtx_flag[ n ][ j – 1 ] )* | |
| ***abs_level_gtx_flag****[ n ][ j ]* | ae(v) |
| *MaxCcbs– –* | |
| *AbsLevelPassX[ xC ][ yC ] += 2 * abs_level_gtx_flag[ n ][ j ]* | |
| *}* | |
| } | |
| iSecondPassBypassPos = n; | |
| /* remainder scan pass */ | |
| for( n = 0; n < iFirstPassBypassPos ; n++ ) { | |
| xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ] | |
| yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ] | |
| minLevel = (scanPos >= iSecondPassBypassPos ? 4 : 10); | |
| if( abs_level_gtx_flag[ n ][ 0 ]) | |
| **par_level_flag**[ n ] | ae(v) |
| *if( abs_level_gtx_flag[ n ][ 4 ] )* | |
| ***abs_remainder****[ n ]* | *ae(v)* |
| TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ] = ( 1 – 2 * coeff_sign_flag[ n ] ) * ( AbsLevelPassX[ xC ][ yC ] + par_level_flag[ n ] + abs_remainder[ n ] ) | |
| } | |
| /* dec_abs_level scan pass */ | |
| *for( n = iFirstPassBypassPos; n <= numSbCoeff – 1; n++ ) {* | |
| *xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ]* | |
| *yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ]* | |
| ***dec_abs_level****[ n ]* | ae(v) |
| *if( sig_coeff_flag[ xC ][ yC ]* | |
| ***coeff_sign_flag****[ n ]* | ae(v) |
| *TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ] = ( 1 – 2 * coeff_sign_flag[ n ] )* dec_abs_level[n] | |
| *}* | |
| *}* | |
| } | |

**FIG. 15 (continued)**

【図１６】



【図１７－１】

| residual_ts_coding( x0, y0, log2TbWidth, log2TbHeight, cIdx ) { | Descriptor |
|---|---|
| log2SbSize = ( Min( log2TbWidth, log2TbHeight ) < 2 ? 1 : 2 ) | |
| numSbCoeff = 1 << ( log2SbSize << 1 ) | |
| lastSubBlock = ( 1 << ( log2TbWidth + log2TbHeight – 2 * log2SbSize ) ) – 1 | |
| inferSbCbf = 1 | |
| MaxCcbs = 2 * ( 1 << log2TbWidth ) * ( 1<< log2TbHeight ) | |
| for( i =0; i <= lastSubBlock; i++ ) { | |
| xS = DiagScanOrder[ log2TbWidth – log2SbSize ][ log2TbHeight – log2SbSize ][ i ][ 0 ] | |
| yS = DiagScanOrder[ log2TbWidth – log2SbSize ][ log2TbHeight – log2SbSize ][ i ][ 1 ] | |
| if( ( i != lastSubBlock \|\| !inferSbCbf ) { | |
| **coded_sub_block_flag**[ xS ][ yS ] | ae(v) |
| } | |
| if( coded_sub_block_flag[ xS ][ yS ] && i < lastSubBlock ) | |
| inferSbCbf = 0 | |
| /* First scan pass */ | |
| inferSbSigCoeffFlag = 1 | |
| for( n = 0; n <= numSbCoeff – 1 **&& *MaxCcbs* >= 8**; n++ ) { | |
| xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ] | |
| yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ] | |
| if( coded_sub_block_flag[ xS ][ yS ] && ( n != numSbCoeff – 1 \|\| !inferSbSigCoeffFlag ) ) { | |
| **sig_coeff_flag**[ xC ][ yC ] | ae(v) |
| MaxCcbs– – | |
| if( sig_coeff_flag[ xC ][ yC ] ) | |
| inferSbSigCoeffFlag = 0 | |
| } | |
| CoeffSignLevel[ xC ][ yC ] = 0 | |
| if( sig_coeff_flag[ xC ][ yC ] ) { | |
| **coeff_sign_flag**[ n ] | ae(v) |
| MaxCcbs– – | |
| CoeffSignLevel[ xC ][ yC ] = ( coeff_sign_flag[ n ] > 0 ? –1 : 1 ) | |
| **abs_level_gtx_flag**[ n ][ 0 ] | ae(v) |
| MaxCcbs– – | |
| if( abs_level_gtx_flag[ n ][ 0 ] ) { | |
| **par_level_flag**[ n ] | ae(v) |
| MaxCcbs– – | |
| } | |
| } | |
| AbsLevelPassX[ xC ][ yC ] = sig_coeff_flag[ xC ][ yC ] + par_level_flag[ n ] + abs_level_gtx_flag[ n ][ 0 ] | |

**FIG. 17**

【図１７－２】

| | |
|---|---|
| ~~}~~ | |
| ~~/* Greater than X scan pass (numGtXFlags=5) */~~ | |
| ~~for( n = 0; n <= numSbCoeff – 1; n++ ) {~~ | |
| ~~xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ]~~ | |
| ~~yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ]~~ | |
| for( j = 1; j < 5; j++ ) { | |
| if( abs_level_gtx_flag[ n ][ j – 1 ] ){ | |
| **abs_level_gtx_flag**[ n ][ j ] | ae(v) |
| MaxCcbs– – | |
| AbsLevelPassX[ xC ][ yC ] += 2 * abs_level_gtx_flag[ n ][ j ] | |
| } | |
| } | |
| } | |
| ***iFirstPassBypassPos = n;*** | |
| /* remainder scan pass */ | |
| for( n = 0; n <= ~~numSbCoeff – 1~~ ***iFirstPassBypassPos***; n++ ) { | |
| xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ] | |
| yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ] | |
| if( abs_level_gtx_flag[ n ][ 4 ] ) | |
| **abs_remainder**[ n ] | ae(v) |
| if( intra_bdpcm_flag == 0 ) { | |
| absRightCoeff = abs( TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC – 1 ][ yC ] ) | |
| absBelowCoeff = abs( TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC – 1 ] ) | |
| predCoeff = Max( absRightCoeff, absBelowCoeff ) | |
| if( AbsLevelPassX[ xC ][ yC ] + abs_remainder[ n ] == 1 && predCoeff > 0 ) | |
| TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ] = ( 1 – 2 * coeff_sign_flag[ n ] ) * predCoeff | |
| else if( AbsLevelPassX[ xC ][ yC ] + abs_remainder[ n ] <= predCoeff ) | |
| TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ] = ( 1 – 2 * coeff_sign_flag[ n ] ) * ( AbsLevelPassX[ xC ][ yC ] + abs_remainder[ n ] – 1) | |
| Else | |
| TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ] = ( 1 – 2 * coeff_sign_flag[ n ] ) * ( AbsLevelPassX[ xC ][ yC ] + abs_remainder[ n ] ) | |
| } else | |
| TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ] = ( 1 – 2 * coeff_sign_flag[ n ] ) * ( AbsLevelPassX[ xC ][ yC ] + abs_remainder[ n ] ) | |
| } | |
| /* *dec_abs_level scan pass* */ | |
| *for( n = iFirstPassBypassPos; n <= numSbCoeff – 1; n++ ) {* | |
| *xC = ( xS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ]* | |
| *yC = ( yS << log2SbSize ) + DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ]* | |

**FIG. 17 (continued)**

【図１７－３】

| | |
|---|---|
| ***dec_abs_level***[ n ] | ***ae(v)*** |
| *if(dec_abs_level[ n ] )* | |
| ***coeff_sign_flag***[ n ] | ***ae(v)*** |
| *if( intra_bdpcm_flag == 0 ) {* | |
| *absRightCoeff = abs( TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC – 1 ][ yC ] )* | |
| *absBelowCoeff = abs( TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC – 1 ] )* | |
| *predCoeff = Max( absRightCoeff, absBelowCoeff )* | |
| *if( AbsLevelPassX[ xC ][ yC ] + abs_remainder[ n ] == 1 && predCoeff > 0 )* | |
| *TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ] = ( 1 – 2 * coeff_sign_flag[ n ] ) * predCoeff* | |
| *else if( AbsLevelPassX[ xC ][ yC ] + abs_remainder[ n ] <= predCoeff )* | |
| *TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ] = ( 1 – 2 * coeff_sign_flag[ n ] ) * ( AbsLevelPassX[ xC ][ yC ] + abs_remainder[ n ] – 1)* | |
| *Else* | |
| *TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ] = ( 1 – 2 * coeff_sign_flag[ n ] ) * ( AbsLevelPassX[ xC ][ yC ] + abs_remainder[ n ] )* | |
| *} else* | |
| *TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ] = ( 1 – 2 * coeff_sign_flag[ n ] ) * ( AbsLevelPassX[ xC ][ yC ] + abs_remainder[ n ] )* | |
| *}* | |
| *}* | |
| } | |

**FIG. 17 (continued)**

【図１８】

| **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 |

**FIG. 18**

フロントページの続き

(33)優先権主張国・地域又は機関
米国(US)
(31)優先権主張番号 62/902,115
(32)優先日 令和1年9月18日(2019.9.18)
(33)優先権主張国・地域又は機関
米国(US)
(72)発明者 サーヴァー，モハメッド，ジー．
アメリカ合衆国，カリフォルニア州 ９４４０２，サン マテオ，サウス エル カミーノ レアル ４００，スイート ４００，アリババ グループ リーガル デパートメント
(72)発明者 イエ，ヤン
アメリカ合衆国，カリフォルニア州 ９４４０２，サン マテオ，サウス エル カミーノ レアル ４００，スイート ４００，アリババ グループ リーガル デパートメント
(72)発明者 ルオ，ジャンコン
アメリカ合衆国，カリフォルニア州 ９４４０２，サン マテオ，サウス エル カミーノ レアル ４００，スイート ４００，アリババ グループ リーガル デパートメント
審査官 久保 光宏
(56)参考文献 Jianle Chen, et al.，"Algorithm description for Versatile Video Coding and Test Model 5 (VTM 5)"，Document: JVET-N1002-v2, [online]，JVET-N1002 (version 2)，Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11，2019年06月11日，Pages 1,4,5,56-59,74，[令和5年1月26日検索], インターネット，〈URL: https://jvet-experts.org/doc_end_user/current_document.php?id=6641〉 and 〈URL: https://jvet-experts.org/doc_end_user/documents/14_Geneva/wg11/JVET-N1002-v2.zip〉.，(See document file "JVET-N1002-v2.docx" in the zip file "JVET-N1002-v2.zip".)
Yusuke Kato, et al.，"Non-CE7: Unification of syntaxes after CCB count exceeds the maximum number between transform residual and transform skip residual"，Document: JVET-O0406-v1, [online]，JVET-O0406 (version 1)，Joint Video Exploration Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11，2019年06月25日，Pages 1-8，[令和7年3月11日検索], インターネット，〈URL: https://jvet-experts.org/doc_end_user/current_document.php?id=7011〉 and 〈URL: https://jvet-experts.org/doc_end_user/documents/15_Gothenburg/wg11/JVET-O0406-v1.zip〉.，(See document file "JVET-O0406-v1.docx" in the zip file "JVET-O0406-v1.zip".)
Yusuke Kato, et al.，"CE7-related: Unification of CCB check method and bypass coding between two residual coding modes"，Document: JVET-P0298-v1, [online]，JVET-P0298 (version 2)，Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11，2019年09月24日，Pages 1-11，[令和7年3月11日検索], インターネット，〈URL: https://jvet-experts.org/doc_end_user/current_document.php?id=8087〉 and 〈URL: https://jvet-experts.org/doc_end_user/documents/16_Geneva/wg11/JVET-P0298-v2.zip〉.，(See document file "JVET-P0298-v1.docx" in the zip file "JVET-P0298-v2.zip".)
大久保 榮 監修，「H.265/HEVC教科書」，初版，日本，株式会社インプレスジャパン，2013年10月21日，第１８５～１８８頁，ISBN: 978-4-8443-3468-2.
(58)調査した分野 (Int.Cl.，ＤＢ名)
Ｈ０４Ｎ１９／００－１９／９８
ＣＳＤＢ（日本国特許庁）
学術文献等データベース（日本国特許庁）
ＩＥＥＥＸｐｌｏｒｅ（ＩＥＥＥ）