

(19)日本国特許庁(JP)

(12)公表特許公報(A)

(11)公表番号

特表2025-502878

(P2025-502878A)

(43)公表日 令和7年1月28日(2025.1.28)

(51)国際特許分類

H 0 4 N 19/70 (2014.01)

F I

H 0 4 N 19/70

テーマコード (参考)

5 C 1 5 9

審査請求 未請求 予備審査請求 未請求 (全68頁)

(21)出願番号 特願2024-541663(P2024-541663)
(86)(22)出願日 令和4年11月8日(2022.11.8)
(85)翻訳文提出日 令和6年7月23日(2024.7.23)
(86)国際出願番号 PCT/US2022/079490
(87)国際公開番号 WO2023/132992
(87)国際公開日 令和5年7月13日(2023.7.13)
(31)優先権主張番号 63/266,615
(32)優先日 令和4年1月10日(2022.1.10)
(33)優先権主張国・地域又は機関
米国(US)
(31)優先権主張番号 63/266,616
(32)優先日 令和4年1月10日(2022.1.10)
(33)優先権主張国・地域又は機関
米国(US)
(31)優先権主張番号 63/266,765

最終頁に続く

(71)出願人 516227559
オッポ広東移動通信有限公司
GUANGDONG OPPO MOBI
LE TELECOMMUNICATI
ONS CORP., LTD.
中華人民共和國広東省東莞市長安鎮烏沙
海浜路18号
No. 18 Haibin Road,
Wusha, Chang'an, Don
gguan, Guangdong 52
3860 China
(74)代理人 100126000
弁理士 岩池 満
(74)代理人 100203105
弁理士 江口 能弘

最終頁に続く

(54)【発明の名称】 ビデオ符号化のための共通制約情報の伝送

(57)【要約】

いくつかの実施例では、ビデオデコーダは、ビデオのビットストリームからビデオを復号する。ビデオデコーダは、ビデオのビットストリームから追加ビットのカウントMを抽出し、追加ビットのカウントMは、ビデオのビットストリームに含まれる追加の共通制約情報(GCI)ビットの数を示す。追加ビットはフラグビットを含み、フラグビットは、ビデオに対して制約されるべき各追加符号化ツールをそれぞれ示し、追加ビットのカウントの期待値は0又は6である。抽出された追加ビットのカウントMが6より大きいと決定された場合、デコーダは、ビットストリーム内の6つのフラグビットの後ろのM-6ビットを抽出する。さらに、デコーダは、抽出されたM-6ビットに依存せず、少なくとも部分的に6つのフラグビットによって各追加符号化ツールにそれぞれ規定された制約に基づいて、ビットストリームの残りの部分をピクチャに復号する。

【選択図】 図6

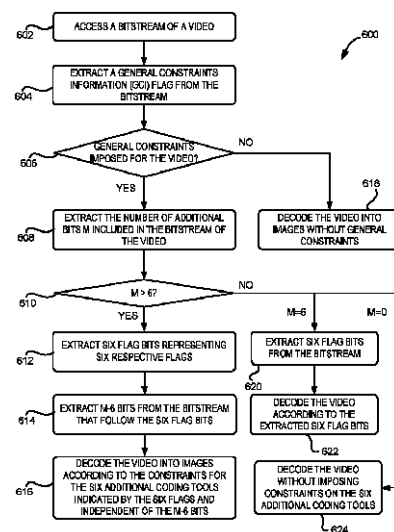


FIG. 6

【特許請求の範囲】

【請求項 1】

ビデオ復号方法であって、

ビデオのビットストリームから追加ビットのカウントMを抽出することであって、前記追加ビットのカウントMは、前記ビデオのビットストリームに含まれる追加の共通制約情報（G C I）ビットの数を示し、前記追加ビットは、フラグビットを含み、前記フラグビットは、前記ビデオに対して制約されるべき各追加符号化ツールを示し、前記追加ビットのカウントの期待値は0又は6である、ことと、

抽出された前記追加ビットのカウントMが6より大きいと決定されたことに応答して、前記ビットストリーム内の6つのフラグビットの後ろのM-6ビットを抽出することと、

抽出された前記M-6ビットに依存せず、少なくとも部分的に前記6つのフラグビットによって各追加符号化ツールにそれぞれ規定された制約に基づいて、前記ビデオのビットストリームの残りの部分をピクチャに復号することと、を含む、ビデオ復号方法。

10

【請求項 2】

前記ビデオ復号方法は、

前記M-6ビットを抽出する前に、前記ビデオのビットストリームから、6つのフラグをそれぞれ表す前記6つのフラグビットを抽出することであって、前記6つのフラグは、前記ビデオに対して制約される6つの追加符号化ツールをそれぞれ示すこと、及び

少なくとも部分的に前記6つのフラグによって前記6つの追加符号化ツールに対して示される制約に基づいて、前記ビデオのビットストリームの残りの部分をピクチャに復号すること、をさらに含む、

20

請求項 1 に記載のビデオ復号方法。

【請求項 3】

前記6つのフラグは、

前記ビデオのピクチャがイントラランダムアクセスポイント（I R A P）ピクチャ又は漸次復号リフレッシュ（G D R）ピクチャであると制限されることを示す第1フラグと、

拡張変換精度を制約しているか否かを示す第2フラグと、

明示的なライスパラメータ伝送を制約しているか否かを示す第3フラグと、

前記ビデオの量子化残差の二値化のための代替ライスパラメータ導出を示す第4フラグと、

30

以前の変換ユニットに基づく二値化のためのライスパラメータ導出を初期化するか否かを示す第5フラグと、

変換ユニットにおける最後の非ゼロレベルの位置を復号するときに、スコープ内の出力層セット（O l s I n S c o p e）内のピクチャに制約をかけるか否かを示す第6フラグと、を含む、

請求項 2 に記載のビデオ復号方法。

【請求項 4】

少なくとも部分的に前記6つのフラグによって前記6つの追加符号化ツールに対して示される制約に基づいて、前記ビデオのビットストリームの残りの部分をピクチャに復号することは、

40

前記第1フラグが1であることに基づいて、1つ又は複数の出力層セット内のすべてのピクチャがp h _ r e c o v e r y _ p o c _ c n tが0に等しいG D Rピクチャ、又はI R A Pピクチャであると決定し、前記1つ又は複数の出力層セット内の前記G D Rピクチャ又は前記I R A Pピクチャを復号すること、

前記第2フラグが1であることに基づいて、拡張変換精度が制約されていると決定し、拡張ダイナミックレンジが使用されないように、前記スコープ内の出力層セット（O l s I n S c o p e）内のピクチャのs p s _ e x t e n d e d _ p r e c i s i o n _ f l a gを0に設定し、前記ビデオのビットストリームの残りの部分を復号すること、

前記第3フラグが1であることに基づいて、明示的なライスパラメータ伝送が制約されていると決定し、前記スコープ内の出力層セット（O l s I n S c o p e）内のピクチャ

50

の代替ライスパラメータ伝送を無効にして、前記ビデオのビットストリームの残りの部分を復号すること、

前記第4フラグが1であることに基づいて、前記ビデオの量子化残差の二値化のための代替ライスパラメータ導出が制約されていると決定し、前記スコープ内の出力層セット（O l s I n S c o p e）内のピクチャの代替ライスパラメータ伝送を無効にして、前記ビデオのビットストリームの残りの部分を復号すること、

前記第5フラグが1であると決定したことに基づいて、以前の変換ユニット状態に基づく二値化のためのライスパラメータ導出の初期化が制約されていると決定し、前記スコープ内の出力層セット（O l s I n S c o p e）内のピクチャに対して以前の変換ユニット状態に基づいてライスパラメータを初期化せずに、前記ビデオのビットストリームの残りの部分を復号すること、又は

10

前記第6フラグが1であることに基づいて、最後の重要係数の座標がスライスの各変換ブロックの左上隅に対して符号化されると決定し、前記最後の重要係数の復号後の座標が前記スライスの各変換ブロックの左上隅に対するものであると解釈して、前記ビデオのビットストリームの残りの部分を復号すること、のうちの1つ又は複数を含む、

請求項3に記載のビデオ復号方法。

【請求項5】

前記ビデオ復号方法は、

前記ビットストリームに前記第1フラグが存在しないと決定し、前記第1フラグの値が0であると推測して、対応する符号化ツールに制約がかけられていないことを示すこと、

20

前記ビットストリームに前記第2フラグが存在しないと決定し、前記第2フラグの値が0であると推測して、対応する符号化ツールに制約がかけられていないことを示すこと、

前記ビットストリームに前記第3フラグが存在しないと決定し、前記第3フラグの値が0であると推測して、対応する符号化ツールに制約がかけられていないことを示すこと、

前記ビットストリームに前記第4フラグが存在しないと決定し、前記第4フラグの値が0であると推測して、対応する符号化ツールに制約がかけられていないことを示すこと、

前記ビットストリームに前記第5フラグが存在しないと決定し、前記第5フラグの値が0であると推測して、対応する符号化ツールに制約がかけられていないことを示すこと、

前記ビットストリームに前記第6フラグが存在しないと決定し、前記第6フラグの値が0であると推測して、対応する符号化ツールに制約がかけられていないことを示すこと、

30

のうちの1つ又は複数をさらに含む、

請求項3に記載のビデオ復号方法。

【請求項6】

前記追加ビットのカウントMを抽出する前に、前記ビデオ復号方法は、

前記ビデオのビットストリームから共通制約情報（G C I）フラグを抽出すること、及び

前記G C Iフラグの値に基づいて、前記ビデオに1つ又は複数の共通制約がかけられていると決定すること、をさらに含む、

請求項1に記載のビデオ復号方法。

【請求項7】

前記G C Iフラグは、前記ビデオのネットワークパケット、前記ビデオのビデオパラメータセット、又は前記ビデオのシーケンスパラメータセットから抽出されるものである、

40

請求項6に記載のビデオ復号方法。

【請求項8】

プログラムコードを記憶した非一時的なコンピュータ可読媒体であって、前記プログラムコードは、1つ又は複数の処理機器に、動作を実行させ、前記動作は、

ビデオのビットストリームから追加ビットのカウントMを抽出することであって、前記追加ビットのカウントMは、前記ビデオのビットストリームに含まれる追加の共通制約情報（G C I）ビットの数を示し、前記追加ビットは、フラグビットを含み、前記フラグビットは、前記ビデオに対して制約されるべき各追加符号化ツールを示し、前記追加ビット

50

のカウン트의期待値は0又は6である、ことと、

抽出された前記追加ビットのカウンツMが6より大きいと決定されたことに応答して、前記ビットストリーム内の6つのフラグビットの後ろのM-6ビットを抽出することと、

抽出された前記M-6ビットに依存せず、少なくとも部分的に前記6つのフラグビットによって各追加符号化ツツにそれぞれ規定された制約に基づいて、前記ビデオのビットストリームの残りの部分をピクチャに復号することと、を含む、非一時的なコンピュータ可読媒体。

【請求項9】

前記動作は、

前記M-6ビットを抽出する前に、前記ビデオのビットストリームから、6つのフラグをそれぞれ表す前記6つのフラグビットを抽出することであって、前記6つのフラグは、前記ビデオに対して制約される6つの追加符号化ツツをそれぞれ示すこと、及び

少なくとも部分的に前記6つのフラグによって前記6つの追加符号化ツツに対して示される制約に基づいて、前記ビデオのビットストリームの残りの部分をピクチャに復号すること、をさらに含む、

請求項8に記載の非一時的なコンピュータ可読媒体。

【請求項10】

前記6つのフラグは、

前記ビデオのピクチャがイントラランダムアクセスポイント（IRAP）ピクチャ又は漸次復号リフレッシュ（GDR）ピクチャであると制限されることを示す第1フラグと、

拡張変換精度を制約しているか否かを示す第2フラグと、

明示的なライスパラメータ伝送を制約しているか否かを示す第3フラグと、

前記ビデオの量子化残差の二値化のための代替ライスパラメータ導出を示す第4フラグと、

以前の変換ユニットに基づく二値化のためのライスパラメータ導出を初期化するか否かを示す第5フラグと、

変換ユニットにおける最後の非ゼロレベルの位置を復号するときに、スコープ内の出力層セット（OlsInScope）内のピクチャに制約をかけるか否かを示す第6フラグと、を含む、

請求項9に記載の非一時的なコンピュータ可読媒体。

【請求項11】

少なくとも部分的に前記6つのフラグによって前記6つの追加符号化ツツに対して示される制約に基づいて、前記ビデオのビットストリームの残りの部分をピクチャに復号することは、

前記第1フラグが1であることに基づいて、1つ又は複数の出力層セット内のすべてのピクチャがph__recovery__poc__cntが0に等しいGDRピクチャ、又はIRAPピクチャであると決定し、前記1つ又は複数の出力層セット内の前記GDRピクチャ又は前記IRAPピクチャを復号すること、

前記第2フラグが1であることに基づいて、拡張変換精度が制約されていると決定し、拡張ダイナミックレンジが使用されないように、前記スコープ内の出力層セット（OlsInScope）内のピクチャのsps__extended__precision__flagを0に設定し、前記ビデオのビットストリームの残りの部分を復号すること、

前記第3フラグが1であることに基づいて、明示的なライスパラメータ伝送が制約されていると決定し、前記スコープ内の出力層セット（OlsInScope）内のピクチャの代替ライスパラメータ伝送を無効にして、前記ビデオのビットストリームの残りの部分を復号すること、

前記第4フラグが1であることに基づいて、前記ビデオの量子化残差の二値化のための代替ライスパラメータ導出が制約されていると決定し、前記スコープ内の出力層セット（OlsInScope）内のピクチャの代替ライスパラメータ伝送を無効にして、前記ビデオのビットストリームの残りの部分を復号すること、

前記第 5 フラグが 1 であると決定したことに基づいて、以前の変換ユニット状態に基づく二値化のためのライスパラメータ導出の初期化が制約されていると決定し、前記スコープ内の出力層セット (O l s I n S c o p e) 内のピクチャに対して以前の変換ユニット状態に基づいてライスパラメータを初期化せずに、前記ビデオのビットストリームの残りの部分を復号すること、又は

前記第 6 フラグが 1 であることに基づいて、最後の重要係数の座標がスライスの各変換ブロックの左上隅に対して符号化されると決定し、前記最後の重要係数の復号後の座標が前記スライスの各変換ブロックの左上隅に対するものであると解釈して、前記ビデオのビットストリームの残りの部分を復号すること、のうちの 1 つ又は複数を含む、

請求項 10 に記載の非一時的なコンピュータ可読媒体。

10

【請求項 12】

前記動作は、

前記ビットストリームに前記第 1 フラグが存在しないと決定し、前記第 1 フラグの値が 0 であると推測して、対応する符号化ツールに制約がかけられていないことを示すこと、

前記ビットストリームに前記第 2 フラグが存在しないと決定し、前記第 2 フラグの値が 0 であると推測して、対応する符号化ツールに制約がかけられていないことを示すこと、

前記ビットストリームに前記第 3 フラグが存在しないと決定し、前記第 3 フラグの値が 0 であると推測して、対応する符号化ツールに制約がかけられていないことを示すこと、

前記ビットストリームに前記第 4 フラグが存在しないと決定し、前記第 4 フラグの値が 0 であると推測して、対応する符号化ツールに制約がかけられていないことを示すこと、

20

前記ビットストリームに前記第 5 フラグが存在しないと決定し、前記第 5 フラグの値が 0 であると推測して、対応する符号化ツールに制約がかけられていないことを示すこと、

前記ビットストリームに前記第 6 フラグが存在しないと決定し、前記第 6 フラグの値が 0 であると推測して、対応する符号化ツールに制約がかけられていないことを示すこと、

のうちの 1 つ又は複数をさらに含む、

請求項 10 に記載の非一時的なコンピュータ可読媒体。

【請求項 13】

前記追加ビットのカウント M を抽出する前に、前記動作は、

前記ビデオのビットストリームから共通制約情報 (G C I) フラグを抽出すること、及び

30

前記 G C I フラグの値に基づいて、前記ビデオに 1 つ又は複数の共通制約がかけていると決定すること、をさらに含む、

請求項 8 に記載の非一時的なコンピュータ可読媒体。

【請求項 14】

前記 G C I フラグは、前記ビデオのネットワークパケット、前記ビデオのビデオパラメータセット、又は前記ビデオのシーケンスパラメータセットから抽出されるものである、

請求項 13 に記載の非一時的なコンピュータ可読媒体。

【請求項 15】

システムであって、

処理機器と、

40

前記処理機器と通信可能に結合された非一時的なコンピュータ可読媒体と、を備え、前記処理機器は、前記非一時的なコンピュータ可読媒体に記憶されたプログラムコードを実行することにより、動作を実行するように構成され、前記動作は、

ビデオのビットストリームから追加ビットのカウント M を抽出することであって、前記追加ビットのカウント M は、前記ビデオのビットストリームに含まれる追加の共通制約情報 (G C I) ビットの数を示し、前記追加ビットは、フラグビットを含み、前記フラグビットは、前記ビデオに対して制約されるべき各追加符号化ツールを示し、前記追加ビットのカウントの期待値は 0 又は 6 である、ことと、

抽出された前記追加ビットのカウント M が 6 より大きいと決定されたことに応答して、前記ビットストリーム内の 6 つのフラグビットの後ろの M - 6 ビットを抽出することと、

50

抽出された前記 M-6 ビットに依存せず、少なくとも部分的に前記 6 つのフラグビットによって各追加符号化ツールにそれぞれ規定された制約に基づいて、前記ビデオのビットストリームの残りの部分をピクチャに復号することと、を含む、システム。

【請求項 16】

前記動作は、

前記 M-6 ビットを抽出する前に、前記ビデオのビットストリームから、6 つのフラグをそれぞれ表す前記 6 つのフラグビットを抽出することであって、前記 6 つのフラグは、前記ビデオに対して制約される 6 つの追加符号化ツールをそれぞれ示すこと、及び

少なくとも部分的に前記 6 つのフラグによって前記 6 つの追加符号化ツールに対して示される制約に基づいて、前記ビデオのビットストリームの残りの部分をピクチャに復号すること、をさらに含む、

請求項 15 に記載のシステム。

【請求項 17】

前記 6 つのフラグは、

前記ビデオのピクチャがイントラランダムアクセスポイント (IRAP) ピクチャ又は漸次復号リフレッシュ (GDR) ピクチャであると制限されることを示す第 1 フラグと、

拡張変換精度を制約しているか否かを示す第 2 フラグと、

明示的なライスパラメータ伝送を制約しているか否かを示す第 3 フラグと、

前記ビデオの量子化残差の二値化のための代替ライスパラメータ導出を示す第 4 フラグと、

以前の変換ユニットに基づく二値化のためのライスパラメータ導出を初期化するか否かを示す第 5 フラグと、

変換ユニットにおける最後の非ゼロレベルの位置を復号するときに、スコープ内の出力層セット (OlsInScope) 内のピクチャに制約をかけるか否かを示す第 6 フラグと、を含む、

請求項 16 に記載のシステム。

【請求項 18】

少なくとも部分的に前記 6 つのフラグによって前記 6 つの追加符号化ツールに対して示される制約に基づいて、前記ビデオのビットストリームの残りの部分をピクチャに復号することは、

前記第 1 フラグが 1 であることに基づいて、1 つ又は複数の出力層セット内のすべてのピクチャが `ph_recovery_poc_cnt` が 0 に等しい GDR ピクチャ、又は IRAP ピクチャであると決定し、前記 1 つ又は複数の出力層セット内の前記 GDR ピクチャ又は前記 IRAP ピクチャを復号すること、

前記第 2 フラグが 1 であることに基づいて、拡張変換精度が制約されていると決定し、拡張ダイナミックレンジが使用されないように、前記スコープ内の出力層セット (OlsInScope) 内のピクチャの `sps_extended_precision_flag` を 0 に設定し、前記ビデオのビットストリームの残りの部分を復号すること、

前記第 3 フラグが 1 であることに基づいて、明示的なライスパラメータ伝送が制約されていると決定し、前記スコープ内の出力層セット (OlsInScope) 内のピクチャの代替ライスパラメータ伝送を無効にして、前記ビデオのビットストリームの残りの部分を復号すること、

前記第 4 フラグが 1 であることに基づいて、前記ビデオの量子化残差の二値化のための代替ライスパラメータ導出が制約されていると決定し、前記スコープ内の出力層セット (OlsInScope) 内のピクチャの代替ライスパラメータ伝送を無効にして、前記ビデオのビットストリームの残りの部分を復号すること、

前記第 5 フラグが 1 であることに基づいて、以前の変換ユニット状態に基づく二値化のためのライスパラメータ導出の初期化が制約されていると決定し、前記スコープ内の出力層セット (OlsInScope) 内のピクチャに対して以前の変換ユニット状態に基づいてライスパラメータを初期化せずに、前記ビデオのビットストリームの残り

10

20

30

40

50

の部分で復号すること、又は

前記第6フラグが1であることに基づいて、最後の重要係数の座標がスライスの各変換ブロックの左上隅に対して符号化されると決定し、前記最後の重要係数の復号後の座標が前記スライスの各変換ブロックの左上隅に対するものであると解釈して、前記ビデオのビットストリームの残りの部分を復号すること、のうちの1つ又は複数を含む、

請求項17に記載のシステム。

【請求項19】

前記動作は、

前記ビットストリームに前記第1フラグが存在しないと決定し、前記第1フラグの値が0であると推測して、対応する符号化ツールに制約がかけられていないことを示すこと、

前記ビットストリームに前記第2フラグが存在しないと決定し、前記第2フラグの値が0であると推測して、対応する符号化ツールに制約がかけられていないことを示すこと、

前記ビットストリームに前記第3フラグが存在しないと決定し、前記第3フラグの値が0であると推測して、対応する符号化ツールに制約がかけられていないことを示すこと、

前記ビットストリームに前記第4フラグが存在しないと決定し、前記第4フラグの値が0であると推測して、対応する符号化ツールに制約がかけられていないことを示すこと、

前記ビットストリームに前記第5フラグが存在しないと決定し、前記第5フラグの値が0であると推測して、対応する符号化ツールに制約がかけられていないことを示すこと、

前記ビットストリームに前記第6フラグが存在しないと決定し、前記第6フラグの値が0であると推測して、対応する符号化ツールに制約がかけられていないことを示すこと、

のうちの1つ又は複数をさらに含む、

請求項17に記載の非一時的なコンピュータ可読媒体。

【請求項20】

前記追加ビットのカウントMを抽出する前に、前記動作は、

前記ビデオのネットワークパケット、前記ビデオのビデオパラメータセット又は前記ビデオのシーケンスパラメータセットから共通制約情報(GCI)フラグを抽出することにより、前記ビデオのビットストリームから前記GCIフラグを抽出すること、及び

前記GCIフラグの値に基づいて、ビデオに1つ又は複数の共通制約がかけられていると決定すること、をさらに含む、

請求項15に記載の非一時的なコンピュータ可読媒体。

【発明の詳細な説明】

【技術分野】

【0001】

(関連出願への相互参照)

本願は、2022年1月10日に提出された、発明の名称が「Signaling Methods for General Constraints Information for Video Coding」である米国仮出願第63/266,615号、2022年1月10日に提出された、発明の名称が「Initialization Method for General Constraint Information Flags for Video Coding」である米国仮出願第63/266,616号、2022年1月13日に提出された、発明の名称が「Signaling and Initialization Methods for General Constraints Information for Video Coding」である米国仮出願第63/266,765号の優先権を主張し、これらの出願のすべての内容は、参照により本願に組み込まれている。

【0002】

本開示は、一般に、ビデオ処理に関する。具体的に、本開示は、ビデオ符号化のための共通制約情報の伝送と初期化に関する。

【背景技術】

【0003】

スマートフォン、タブレットコンピュータ、コンピュータなどのカメラ付き機器の普及により、ビデオ又はピクチャの撮影がかつてないほど簡単になっている。しかし、ショートビデオであっても、そのデータ量はかなり大きい可能性がある。ビデオ符号化技術（ビデオ符号化とビデオ復号を含む）により、ビデオデータをより小さなサイズに圧縮でき、それにより、様々なビデオを記憶及び伝送することができる。ビデオ符号化は、デジタルテレビ放送、インターネットやモバイルネットワークのビデオ伝送、リアルタイムアプリケーション（例えば、ビデオチャット、ビデオ会議など）、DVD、ブルーレイディスクなど、様々なアプリケーションに幅広く適用されている。ビデオを記憶するための記憶容量及び／又はビデオを伝送するためのネットワーク帯域幅の消費を削減するために、ビデオ符号化スキームの効率を向上させることが期待されている。

10

【発明の概要】

【0004】

いくつかの実施例は、ビデオ符号化のための共通制約情報の伝送と初期化に係る。一例では、ビデオ復号方法は、ビデオのビットストリームから追加ビットのカウントMを抽出することであって、追加ビットのカウントMは、ビデオのビットストリームに含まれる追加の共通制約情報（GCI）ビットの数を示し、追加ビットはフラグビットを含み、フラグビットは、当該ビデオに対して制約される各追加符号化ツールをそれぞれ示し、追加ビットのカウントの期待値は0又は6である、ことと、抽出された追加ビットのカウントMが6より大きいと決定された場合、ビットストリーム内の6つのフラグビットの後ろのM-6ビットを抽出することと、抽出されたM-6ビットに依存せず、少なくとも部分的に当該6つのフラグビットによって各追加符号化ツールに規定された制約に基づいて、ビデオのビットストリームの残りの部分をピクチャに復号することと、を含む。

20

【0005】

別の例では、非一時的なコンピュータ可読媒体には、プログラムコードが記憶されており、プログラムコードは、1つ又は複数の処理機器に動作を実行させることができる。動作は、ビデオのビットストリームから追加ビットのカウントMを抽出することであって、追加ビットのカウントMは、ビデオのビットストリームに含まれる追加の共通制約情報（GCI）ビットの数を示し、追加ビットはフラグビットを含み、フラグビットは、当該ビデオに対して制約される各追加符号化ツールをそれぞれ示し、追加ビットのカウントの期待値は0又は6である、ことと、抽出された追加ビットのカウントMが6より大きいと決定された場合、ビットストリーム内の6つのフラグビットの後ろのM-6ビットを抽出することと、抽出されたM-6ビットに依存せず、少なくとも部分的に当該6つのフラグビットによって各追加符号化ツールに規定された制約に基づいて、ビデオのビットストリームの残りの部分をピクチャに復号することと、を含む。

30

【0006】

さらに別の例では、システムは、処理機器と、当該処理機器と通信可能に結合された非一時的なコンピュータ可読媒体と、を備える。当該処理機器は、非一時的なコンピュータ可読媒体に記憶されたプログラムコードを実行することにより、動作を実行するように構成される。動作は、ビデオのビットストリームから追加ビットのカウントMを抽出することであって、追加ビットのカウントMは、ビデオのビットストリームに含まれる追加の共通制約情報（GCI）ビットの数を示し、追加ビットはフラグビットを含み、フラグビットは、当該ビデオに対して制約される各追加符号化ツールをそれぞれ示し、追加ビットのカウントの期待値は0又は6である、ことと、抽出された追加ビットのカウントMが6より大きいと決定された場合、ビットストリーム内の6つのフラグビットの後ろのM-6ビットを抽出することと、抽出されたM-6ビットに依存せず、少なくとも部分的に当該6つのフラグビットによって各追加符号化ツールに規定された制約に基づいて、ビデオのビットストリームの残りの部分をピクチャに復号することと、を含む。

40

【0007】

これらの例示的な実施例は、本開示を限定又は定義するのではなく、本開示の理解を助けるための例を提供することを意図するものである。追加の実施例については、詳細な説

50

明で説明されており、そこでさらに詳しく説明されている。

【図面の簡単な説明】

【0008】

【図1】本明細書提供された実施例を実現するように構成されたビデオエンコーダの一例を示すブロック図である。

【図2】本明細書提供された実施例を実現するように構成されたビデオデコーダの一例を示すブロック図である。

【図3】本開示のいくつかの実施例に係る、ビデオ内のピクチャの符号化ツリーユニット分割の一例を示す。

【図4】本開示のいくつかの実施例に係る、符号化ツリーユニットの符号化ユニット分割の例を示す。

【図5】本開示のいくつかの実施例に係る、ビデオ復号プロセスの例を示す。

【図6】本開示のいくつかの実施例に係る、ビデオ復号プロセスの別の例を示す。

【図7】本開示のいくつかの実施例に係る、ビデオ復号プロセスの別の例を示す。

【図8】本開示のいくつかの実施例を実現するためのコンピューティングシステムの一例を示す。

【発明を実施するための形態】

【0009】

図面を参照して以下の詳細な説明を読むことによって、本開示の特徴、実施例、及び利点をよりよく理解することができる。

【0010】

各実施例は、ビデオ符号化のための共通制約情報の伝送と初期化を提供する。上述したように、ますます多くのビデオデータが生成、記憶、伝送されている。ビデオ符号化技術の効率を高めるだけでなく、デコーダ側でビデオ信号を正常に復号できるように、ビデオ符号化の安定性を高めることも有益である。ビデオ復号の安定性に関する問題には、非互換性と不整合が含まれる。ビデオ符号化技術の発展に伴い、新しいビデオ符号化標準も生まれてきた。そのうち、ビデオ符号化標準の1つは、多用途ビデオ符号化標準のバージョン1であり、国際標準化機構（ISO）によって「ISO/IEC 23090-3:2021 情報技術—イマージングメディアの符号化表示—パート3：多用途ビデオ符号化」として、国際電気通信連合（ITU）によって「ITU-T H.26 勧告（08/2020）：多用途ビデオ符号化」として共同で発行される。本開示では、多用途ビデオ符号化標準のバージョン1は、「VVCバージョン1」又は「VVCv1」とも呼ばれる。VVCバージョン1は、多用途ビデオ符号化標準のバージョン2に取って代わり、バージョン2は、ISOによって「ISO/IEC 23090-3:2022 情報技術—イマージングメディアの符号化表示—パート3：多用途ビデオ符号化」として、ITUによって「ITU-T H.266 勧告（04/2022）：多用途ビデオ符号化」として共同で発行される。本開示では、多用途ビデオ符号化標準のバージョン2は、「VVCバージョン2」又は「VVCv2」とも呼ばれる。以前のバージョンのビデオ符号化標準を採用して符号化されたビデオ信号を、新しいバージョンのビデオ符号化標準のビデオデコーダによって正常に復号させるために、ビデオ符号化スキームを、以前のバージョンの符号化標準と後方互換性があるように設計する必要がある。しかし、VVCバージョン2の現在のドラフトで使用されている共通制約情報の伝送に、ビデオ復号が同期できないことを引き起こし、これは、異なるバージョンのビデオ符号化標準間での酷い非互換性の問題である。さらに、VVCバージョン2の現在のドラフトでは、共通制約情報に関連する共通制約フラグがある場合では定義されていない可能性があり、それによってデコーダ実現方式に不明確と不整合が生じる場合がある。本明細書で説明される様々な実施例は、ビデオ符号化のための共通制約情報の伝送と初期化方法を導入してこれらの問題を解決することによって、ビデオ符号化の安定性を高める。

【0011】

VVC標準では、共通制約情報（GCI）構文構造 `general_constraint`

10

20

30

40

50

`nts__info()` は、ビットストリームの特定の制約プロパティを示すために使用される。`GCI` は、制約フラグと非フラグ構文要素のリストを含む。バイナリフラグ `gci__present__flag` は、`GCI` 構文要素が存在するか否かを規定するために使用される。いくつかの実施例において、`VVC` バージョン 2 のビットストリームで共通制約情報を伝送し（即ち、`gci__present__flag` の値が 1 である）、`VVC` バージョン 2 の共通制約情報が N 個の制約される可能性のある追加符号化ツールで構成される場合、これらの N 個の追加符号化ツールに対応する構文要素 `gci__num__additional__bits` の値は、0 又は N のみに設定できる。`gci__num__additional__bits` が 0 に設定される場合、これらの N 個の追加符号化ツールの共通制約フラグは伝送されない。`gci__num__additional__bits` が N に設定される場合、ビットストリームの次の N ビットは、これらの N 個の追加符号化ツールの共通制約フラグを伝送するために使用される。1 つの例では、 N は 6 に設定される。

10

【0012】

いくつかの例では、`VVC` バージョン 2 のビットストリームについて、`gci__num__additional__bits` を 0 又は N 以外の値に設定することが許可されない。ただし、`VVC` バージョン 2 のデコーダは、依然として、`gci__num__additional__bits` を 0 又は N 以外の値に設定する共通制約情報を処理できる。例えば、`gci__num__additional__bits` を M に設定することができ、 M は、0 より大きく N より小さく、又は M は N より大きい。 M が 0 より大きく、且つ N より小さい場合、`gci__num__additional__bits` 構文要素を復号した後、デコーダは、ビットストリームから M ビットを抽出して破棄する。 M が N より大きい場合、`gci__num__additional__bits` 構文要素を復号した後、デコーダは、ビットストリームから N ビットを抽出し、これらの N ビットを、 N 個の追加符号化ツールの共通制約フラグとして解釈する。次に、デコーダは、ビットストリームから $(M - N)$ ビットをさらに抽出して破棄する。他の例では、`VVC` バージョン 2 のデコーダは、`gci__num__additional__bits` を 0 より大きく、且つ N より小さい値に設定する共通制約情報を処理する必要はない。正当な `VVC` バージョン 2 ビットストリームは、`gci__num__reserved__bits` の値を 0 又は N のみに設定できる。`VVC` の未来バージョンのビットストリームでは、`gci__num__additional__bits` を 0 から N 間の値に設定することが許可されない。

20

30

【0013】

いくつかの実施例において、共通制約情報フラグを初期化することは、上述したデコーダ実現の不明確と不整合の問題を解決するために使用される。これらの実施例では、`gci__present__flag` が 1 に等しくかつ `gci__num__additional__bits` が 0 に等しい場合、`general__constraints__info()` は、共通制約情報フラグに関連する符号化ツールに制約をかけない。フラグ値が 0 で制約がないことを示す例では、フラグが存在しない場合、共通制約情報フラグの値が 0 に等しいと推測される。

【0014】

本開示に記載される実施例は、`VVC` バージョン 2 の追加符号化ツールの共通制約フラグの伝送と推測の方法を提供する。先行技術とは異なり、本開示に記載される方法によって生成される高レベル構文ビットストリームは、`VVC` バージョン 1 のデコーダと互換性があり、`VVC` バージョン 1 のデコーダの動作と `VVC` バージョン 2 のデコーダの動作が同期できない状況であっても復号できる。本開示に記載される推理ルールは、`VVC` バージョン 2 の `GCI` 構文要素の `VVC` バージョン 2 の復号動作の不明確を除去する。これらの技術は、様々なビデオ符号化標準において効果的な符号化ツールとなり得る。

40

【0015】

ここで図面を参照すると、図 1 は、ビデオエンコーダ 100 の例を示すブロック図であり、当該ビデオエンコーダ 100 は、本明細書で提出される実施例を実現するように構成される。図 1 に示す例では、ビデオエンコーダ 100 は、分割モジュール 112、変換モ

50

ジュール 114、量子化モジュール 115、逆量子化モジュール 118、逆変換モジュール 119、ループ内フィルタモジュール 120、イントラ予測モジュール 126、インター予測モジュール 124、動き推定モジュール 122、復号ピクチャバッファ 130、及びエントロピ符号化モジュール 116 を備える。

【0016】

ビデオエンコーダ 100 への入力が入力ビデオ 102 であり、その中には、一連のピクチャ（フレーム又はピクチャとも呼ばれる）が含まれる。ブロックベースのビデオエンコーダでは、各ピクチャに対して、ビデオエンコーダ 100 は、分割モジュール 112 を採用してピクチャをブロック 104 に分割し、各ブロックは、複数の画素を含む。これらのブロックは、マクロブロック、符号化ツリーユニット、符号化ユニット、予測ユニット、及び／又は予測ブロックであってもよい。1 枚のピクチャは、異なるサイズのブロックを含んでもよく、ビデオの異なるピクチャのブロック分割は、異なってもよい。各ブロックは、イントラ予測又はインター予測又はイントラとインターのハイブリット予測などの、異なる予測を使用して符号化されることができる。

【0017】

通常、ビデオ信号の 1 枚目のピクチャは、イントラ符号化されたピクチャであり、イントラ予測のみを使用して符号化される。イントラ予測モードでは、同一ピクチャの符号化済みのデータのみを用いてピクチャのブロックを予測する。イントラ符号化されたピクチャは、他のピクチャからの情報を使用せずに復号することができる。イントラ予測を実行するために、図 1 に示すビデオエンコーダ 100 は、イントラ予測モジュール 126 を使用することができる。イントラ予測モジュール 126 は、同一ピクチャの隣接ブロックの再構成ブロック 136 内の再構成サンプルを使用して、イントラ予測ブロック（予測ブロック 134）を生成するように構成される。ブロックに対して選択されたイントラ予測モードに従ってイントラ予測を実行する。次に、ビデオエンコーダ 100 は、ブロック 104 とイントラ予測ブロック 134 との差を計算する。この差は、残差ブロック 106 と呼ばれる。

【0018】

ブロックから冗長をさらに除去するために、変換モジュール 114 は、ブロック内のサンプルに変換を行うことによって、残差ブロック 106 を変換ドメインに変換する。変換の例は、離散コサイン変換（DCT）又は離散サイン変換（DST）を含むが、これらに限定されない。変換後の値は、変換係数とも呼ばれ、変換ドメインにおける残差ブロックを表す。いくつかの例では、残差ブロックは、変換モジュール 114 の変換を経由せずに直接量子化されることができる。このモードは、変換スキップモードと呼ばれる。

【0019】

ビデオエンコーダ 100 は、量子化モジュール 115 をさらに使用して変換係数を量子化し、量子化係数を得ることができる。量子化は、サンプルを量子化ステップサイズで除算し、その後丸めを行うことを含み、逆量子化は、量子化された値に量子化ステップサイズを乗算することである。このような量子化プロセスは、スカラ量子化と呼ばれる。量子化は、ビデオサンプル（変換済み又は未変換）のダイナミックレンジを縮小して、ビデオサンプルを表すために使用されるビット数を減らすために使用される。

【0020】

ブロック内の係数／サンプルの量子化は、独立して遂行でき、いくつかの既存のビデオ圧縮標準（例えば、H.264 や HEVC など）ではこの量子化方法を使用している。N×M ブロックに対して、係数の量子化と符号化のために、何らかのスキャン順序でブロックの 2D 係数を 1-D アレイに変換することができる。ブロック内係数の量子化は、スキャン順序情報を利用することができる。例えば、ブロック内の所定係数の量子化は、スキャン順序における前の 1 つの量子化値の状態によって決定される可能性がある。符号化効率をさらに向上させるために、複数の量子化器を使用することができる。現在の係数を量子化するためにどの量子化器を使用するかは、符号化／復号スキャン順序における現在の係数の前の情報に依存する。このような量子化方法は、依存量子化と呼ばれる。

10

20

30

40

50

【0021】

量子化ステップサイズを使用して量子化の程度を調整することができる。例えば、スカラー量子化の場合、異なる量子化ステップサイズを使用して、より細かい又はより粗い量子化を実現することができる。量子化ステップサイズが小さいほど、量子化は細くなり、量子化ステップサイズが大きいほど、量子化は粗くなる。量子化ステップサイズは、量子化パラメータ（QP）によって指示することができる。量子化パラメータがビデオの符号化ビットストリームで提供されるため、ビデオデコーダは、復号のために量子化パラメータをアクセスして使用することができる。

【0022】

その後、エントロピ符号化モジュール116は量子化サンプルを符号化して、ビデオ信号のサイズをさらに縮小する。エントロピ符号化モジュール116は、エントロピ符号化アルゴリズムを量子化サンプルに適用するように構成される。いくつかの例では、量子化サンプルは、バイナリbinに二値化され、符号化アルゴリズムによって、当該バイナリbinをさらにビットに圧縮する。二値化方法の例は、ランケーテッドドライブ（TR）と制限付きk次指数ゴロム（EGk）二値化の組み合わせ、及びk次指数ゴロム二値化などを含むが、これらに限定されない。エントロピ符号化アルゴリズムの例は、可変長符号化（VLC：variable length coding）スキーム、コンテキスト自己適応型VLCスキーム（CAVLC：context adaptive VLC）、算術コーディングスキーム、二値化、コンテキスト自己適応型バイナリ算術コーディング（CABAC：context-adaptive binary arithmetic coding）、シンタックスベースのコンテキスト自己適応型バイナリ算術コーディング（SBAC：syntax-based context-adaptive binary arithmetic coding）、確率間隔分割エントロピ（PIPE：probability interval partitioning entropy）符号化又は他のエントロピ符号化技術を含むが、これらに限定されない。エントロピ符号化されたデータは、符号化済みのビデオ132を出力するビットストリームに追加される。

【0023】

上述したように、隣接ブロックからの再構成ブロック136は、ピクチャのブロックのイントラ予測に使用される。1つのブロックの再構成ブロック136の生成には、当該ブロックの再構成残差を計算する必要がある。再構成残差は、ブロックの量子化残差に逆量子化及び逆変換を適用することによって決定されることができる。逆量子化モジュール118は、量子化サンプルに逆量子化を適用して、逆量子化係数を取得するように構成される。逆量子化モジュール118は、量子化モジュール115と同じ量子化ステップサイズを使用して、量子化モジュール115に適用された量子化スキームの逆スキームを適用する。逆変換モジュール119は、変換モジュール114に適用された変換の逆変換（逆DCT又は逆DSTなど）を逆量子化サンプルに適用するように構成される。逆変換モジュール119の出力は、画素ドメインにおけるブロックの再構成残差である。再構成残差をブロックの予測ブロック134に追加することによって、画素ドメインにおける再構成ブロック136を取得することができる。変換がスキップされたブロックに、逆変換モジュール119は適用されない。逆量子化サンプルは、ブロックの再構成残差である。

【0024】

一番目のイントラ予測ピクチャの後ろの後続のピクチャのブロックは、インター予測又はイントラ予測を使用して符号化されることができる。インター予測では、ピクチャ内のブロック予測は、1つ又は複数の以前に符号化されたビデオピクチャに基づいて行われる。インター予測を実行するために、ビデオエンコーダ100は、インター予測モジュール124を使用する。インター予測モジュール124は、動き推定モジュール122によって提供される動き推定に基づいて、ブロックに対して動き補償を行うように構成される。

【0025】

動き推定モジュール122は、現在のピクチャの現在ブロック104を復号参照ピクチ

10

20

30

40

50

ャ 1 0 8 と比較して動き推定を行う。復号参照ピクチャ 1 0 8 は、復号ピクチャバッファ 1 3 0 に記憶される。動き推定モジュール 1 2 2 は、復号参照ピクチャ 1 0 8 から、現在ブロックに最も合致する参照ブロックを選択する。動き推定モジュール 1 2 2 はさらに、参照ブロックの位置（例えば、x、y 座標）と現在ブロックの位置との間のオフセットを識別する。当該オフセットは、動きベクトル（MV）と呼ばれ、選択された参照ブロックと共にインター予測モジュール 1 2 4 に提供される。場合によっては、複数の復号参照ピクチャ 1 0 8 における現在ブロックに対して複数の参照ブロックが識別される。したがって、複数の動きベクトルが生成され、対応する参照ブロックと共にインター予測モジュール 1 2 4 に提供される。

【0026】

インター予測モジュール 1 2 4 は、動きベクトル及び他のインター予測パラメータを使用して動き補償を実行して、現在ブロックの予測、即ち、インター予測ブロック 1 3 4 を生成する。例えば、インター予測モジュール 1 2 4 は、動きベクトルに基づいて、対応する参照ピクチャ内において動きベクトルによって指し示される予測ブロックを特定することができる。複数の予測ブロックが存在する場合、これらの予測ブロックを一定の重みで組み合わせて、現在ブロックの予測ブロック 1 3 4 を生成する。

【0027】

インター予測ブロックの場合、ビデオエンコーダ 1 0 0 は、ブロック 1 0 4 からインター予測ブロック 1 3 4 を減算して残差ブロック 1 0 6 を生成することができる。上述したイントラ予測ブロックの残差と同じ方式で残差ブロック 1 0 6 に対して変換、量子化及びエントロピ符号化を行うことができる。同様に、残差に対して逆量子化、逆変換を行い、そして、対応する予測ブロック 1 3 4 と組み合わせて、インター予測ブロックの再構成ブロック 1 3 6 を得ることができる。

【0028】

動き推定に使用される復号ピクチャ 1 0 8 を得るために、ループ内フィルタモジュール 1 2 0 は、再構成ブロック 1 3 6 を処理する。ループ内フィルタモジュール 1 2 0 は、画素過渡を平滑化するように構成され、それによってビデオ品質が向上する。ループ内フィルタモジュール 1 2 0 は、ディブロックフィルタ、サンプル自己適応オフセット（SAO）フィルタ、自己適応ループフィルタ（ALF）などの、1 つ又は複数のループ内フィルタを実現するように構成される。

【0029】

図 2 は、ビデオデコーダ 2 0 0 の例を示し、当該ビデオデコーダ 2 0 0 は、本明細書で提出される実施例を実現するように構成される。ビデオデコーダ 2 0 0 は、ビットストリームにおける符号化されたビデオ 2 0 2 を処理し、復号ピクチャ 2 0 8 を生成する。図 2 に示す例では、ビデオデコーダ 2 0 0 は、エントロピ復号モジュール 2 1 6、逆量子化モジュール 2 1 8、逆変換モジュール 2 1 9、ループ内フィルタモジュール 2 2 0、イントラ予測モジュール 2 2 6、インター予測モジュール 2 2 4 及び復号ピクチャバッファ 2 3 0 を備える。

【0030】

エントロピ復号モジュール 2 1 6 は、符号化されたビデオ 2 0 2 に対してエントロピ復号を行うように構成される。エントロピ復号モジュール 2 1 6 は、量子化係数、符号化パラメータ（イントラ予測パラメータ及びインター予測パラメータを含む）、及び他の情報を復号する。いくつかの例では、エントロピ復号モジュール 2 1 6 は、符号化されたビデオ 2 0 2 のビットストリームをバイナリ表現に復号し、その後、バイナリ表現を係数の量子化レベルに変換する。エントロピ復号された係数レベルは、逆量子化モジュール 2 1 8 によって逆量子化され、続いて、逆変換モジュール 2 1 9 によって画素ドメインに逆変換される。逆量子化モジュール 2 1 8 と逆変換モジュール 2 1 9 の機能はそれぞれ、上記の図 1 に対して説明された逆量子化モジュール 1 1 8 と逆変換モジュール 1 1 9 と類似している。逆変換された残差ブロックは、再構成ブロック 2 3 6 を生成するために、対応する予測ブロック 2 3 4 に追加されることができる。変換がスキップされたブロックに、逆変

10

20

30

40

50

換モジュール 219 は適用されない。逆量子化モジュール 118 によって生成された逆量子化サンプルは、再構成ブロック 236 を生成するために使用される。

【0031】

特定のブロックの予測ブロック 234 は、当該ブロックの予測モードに基づいて生成される。当該ブロックの符号化パラメータが、当該ブロックがイントラ予測されることを示す場合、同じピクチャ内の参照ブロックの再構成ブロック 236 をイントラ予測モジュール 226 に入力して、当該ブロックの予測ブロック 234 を生成することができる。当該ブロックの符号化パラメータが、当該ブロックがイントラ予測されることを示す場合、予測ブロック 234 は、インター予測モジュール 224 によって生成される。イントラ予測モジュール 226 とインター予測モジュール 224 の機能は、図 1 のイントラ予測モジュール 126 とインター予測モジュール 124 と類似している。

10

【0032】

上記の図 1 に関する記載のように、インター予測には、1 つ又は複数の参照ピクチャが関係する。ビデオデコーダ 200 は、ループ内フィルタモジュール 220 を参照ピクチャの再構成ブロックに適用することにより、参照ピクチャの復号ピクチャ 208 を生成する。復号されたピクチャ 208 は、復号ピクチャバッファ 230 に記憶され、インター予測モジュール 224 に提供して出力のために使用される。

【0033】

ここで図 3 を参照すると、図 3 は、本開示のいくつかの実施例に係る、ビデオ内のピクチャの符号化ツリーユニット分割の一例を示す。図 1 及び図 2 に関する記載のように、ビデオのピクチャを符号化するために、ピクチャは、図 3 に示すように、VVC における符号化ツリーユニット (CTU) 302 のようなブロックに分割される。例えば、CTU 302 は、 128×128 画素のブロックであってもよい。CTU は、順序 (例えば、図 3 に示す順序) に従って処理される。いくつかの例において、図 4 に示すように、ピクチャ内の各 CTU 302 は、1 つ又は複数の符号化ユニット (CU) 402 に分割されることができ、CU 402 はさらに、予測及び変換のために、予測ユニット又は変換ユニット (TU) に分割されることができる。符号化スキームに応じて、CTU 302 は、異なる方式で CU 402 に分割されることができる。例えば、VVC では、CU 402 は、長方形又は正方形であってもよく、且つ予測ユニット又は変換ユニットにさらに分割することなく符号化されることができる。各 CU 402 は、そのルート CTU 302 と同じサイズであってもよいし、ルート CTU 302 の 4×4 ブロックほど小さい細分であってもよい。図 4 に示すように、VVC で CTU 302 を CU 402 に分割する方法としては、四分木分割、二分木分割、三分木分割などがある。図 4 において、実線は四分木分割を示し、破線は二分木又は三分木分割を示す。

20

30

【0034】

VVC バージョン 1 (VVCv1) における共通制約情報について

VVC バージョン 1 では、共通制約情報 (GCI) 構文構造 `general_constraints_info()` は、ビットストリームの特定の制約プロパティを示すために使用される。GCI は、制約フラグと非フラグ構文要素のリストを含む。バイナリフラグ `gci_present_flag` は、GCI 構文要素が存在するか否かを規定するために使用される。`gci_present_flag` が 1 に等しい場合、`general_constraints_info()` 構文構造に GCI 構文要素が存在することを規定する。`gci_present_flag` が 0 に等しい場合、`general_constraints_info()` 構文構造に GCI フィールドが存在せず、且つ `general_constraint_info()` 構文構造にいかなる制約もかけないことを規定する。

40

【0035】

複数のコンテキストにおいて、高レベル構文で共通制約情報を伝送することができる。例えば、復号能力情報のみを含むネットワークパケット、例えば、`nal_unit_type` が 13 に設定された (即ち、DCI_NUT を `nal_unit_type` の名称

50

とした)NAL (Network Abstraction Layer) パケットでGCIを伝送することができ、当該ネットワークパケットは、復号能力情報のみを含む。あるいは、ビデオパラメータセット又はシーケンスパラメータセットでGCIを伝送することができる。

【0036】

GCI 構文構造の目的は、ビットストリームの復号に必要な機能に関する構成情報の検出を可能にし、相互運用点 (IOP: interoperability point) を伝送できるようにすることであり、前記相互運用点でかけられる制限は、プロファイル、ティア、及びレベル (PTL) の規定を超え、その粒度は、以前のビデオ符号化標準で許可されている粒度よりも細かい。サブプロファイルと同様に、GCI 構文構造の使用は、VVCPロファイルのすべての機能をサポートしていないが、特定のアプリケーションのニーズを満たすデコーダの実現方式のために相互運用性を定義することを可能にすることができる。デコーダの実現方式では、GCI 構文要素を調べて、ビットストリームが特定の機能の使用を回避しているか否かをチェックし、復号プロセスをどのように構成するかを決定し、ビットストリームがデコーダによって復号可能か否かを識別することができる。VVCPロファイルのすべての機能をサポートするデコーダの実現方式では、GCI 構文要素値を無視することができ、なぜならば、このようなデコーダは、指示されたPTLに満たすすべてのビットストリームを復号できるためである。

10

【0037】

VVCバージョン1で規定された共通制約情報構文構造は、次のように定義される。

20

【0038】

30

40

50

【表 1 (1) 】

general_constraints_info() {	記述子
gci_present_flag	u(1)
if(gci_present_flag) {	
/* 共通 (general) */	
gci_intra_only_constraint_flag	u(1)
gci_all_layers_independent_constraint_flag	u(1)
gci_one_au_only_constraint_flag	u(1)
/*ピクチャフォーマット(picture format)*/	
gci_sixteen_minus_max_bitdepth_constraint_idc	u(4)
gci_three_minus_max_chroma_format_constraint_idc	u(2)
/* NALユニットタイプに関する (NAL unit type related) */	
gci_no_mixed_nalu_types_in_pic_constraint_flag	u(1)
gci_no_trail_constraint_flag	u(1)
gci_no_stsa_constraint_flag	u(1)
gci_no_rasl_constraint_flag	u(1)
gci_no_radl_constraint_flag	u(1)
gci_no_idr_constraint_flag	u(1)
gci_no_cra_constraint_flag	u(1)
gci_no_gdr_constraint_flag	u(1)
gci_no_aps_constraint_flag	u(1)
gci_no_idr_rpl_constraint_flag	u(1)
/*タイル、スライス、サブピクチャの分割 (tile, slice, subpicture partitioning)*/	
gci_one_tile_per_pic_constraint_flag	u(1)
gci_pic_header_in_slice_header_constraint_flag	u(1)
gci_one_slice_per_pic_constraint_flag	u(1)

10

20

30

40

50

【表 1 (2)】

gci_no_rectangular_slice_constraint_flag	u(1)
gci_one_slice_per_subpic_constraint_flag	u(1)
gci_no_subpic_info_constraint_flag	u(1)
/* CTUとブロックの分割 (CTU and block partitioning) */	
gci_three_minus_max_log2_ctu_size_constraint_idc	u(2)
gci_no_partition_constraints_override_constraint_flag	u(1)
gci_no_mtt_constraint_flag	u(1)
gci_no_qtbtt_dual_tree_intra_constraint_flag	u(1)
/* イントラ (intra) */	
gci_no_palette_constraint_flag	u(1)
gci_no_ibc_constraint_flag	u(1)
gci_no_isp_constraint_flag	u(1)
gci_no_mrl_constraint_flag	u(1)
gci_no_mip_constraint_flag	u(1)
gci_no_cclm_constraint_flag	u(1)
/* インター (inter) */	
gci_no_ref_pic_resampling_constraint_flag	u(1)
gci_no_res_change_in_clvs_constraint_flag	u(1)
gci_no_weighted_prediction_constraint_flag	u(1)
gci_no_ref_wraparound_constraint_flag	u(1)
gci_no_temporal_mvp_constraint_flag	u(1)
gci_no_sbtmvp_constraint_flag	u(1)
gci_no_amvr_constraint_flag	u(1)
gci_no_bdof_constraint_flag	u(1)
gci_no_smvd_constraint_flag	u(1)
gci_no_dmvr_constraint_flag	u(1)

10

20

30

40

50

【表 1 (3) 】

gci_no_mmvd_constraint_flag	u(1)
gci_no_affine_motion_constraint_flag	u(1)
gci_no_prof_constraint_flag	u(1)
gci_no_bcw_constraint_flag	u(1)
gci_no_ciip_constraint_flag	u(1)
gci_no_gpm_constraint_flag	u(1)
/*変換、量子化、残差 (transform, quantization, residual) */	
gci_no_luma_transform_size_64_constraint_flag	u(1)
gci_no_transform_skip_constraint_flag	u(1)
gci_no_bdpcm_constraint_flag	u(1)
gci_no_mts_constraint_flag	u(1)
gci_no_lfnst_constraint_flag	u(1)
gci_no_joint_cbr_constraint_flag	u(1)
gci_no_sbt_constraint_flag	u(1)
gci_no_act_constraint_flag	u(1)
gci_no_explicit_scaling_list_constraint_flag	u(1)
gci_no_dep_quant_constraint_flag	u(1)
gci_no_sign_data_hiding_constraint_flag	u(1)
gci_no_cu_qp_delta_constraint_flag	u(1)
gci_no_chroma_qp_offset_constraint_flag	u(1)
/* ループ内フィルタloop filter */	
gci_no_sao_constraint_flag	u(1)
gci_no_alf_constraint_flag	u(1)
gci_no_ccalf_constraint_flag	u(1)
gci_no_lmcs_constraint_flag	u(1)
gci_no_ladf_constraint_flag	u(1)

10

20

30

40

50

【表 1 (4)】

<code>gci_no_virtual_boundaries_constraint_flag</code>	<code>u(1)</code>
<code>gci_num_reserved_bits</code>	<code>u(8)</code>
<code>for(i = 0; i < gci_num_reserved_bits; i++)</code>	
<code> gci_reserved_zero_bit[i]</code>	<code>u(1)</code>
<code>}</code>	
<code>while(!byte_aligned())</code>	
<code>gci_alignment_zero_bit</code>	<code>f(1)</code>
<code>}</code>	

10

【 0 0 3 9】

共通制約フラグの存在は、`gci_present_flag`の値に決定される。`gci_present_flag`の値が1である場合、ビットストリームに共通制約フラグが存在する。`gci_present_flag`の値が0である場合、ビットストリームに共通制約フラグが存在しない。

20

【 0 0 4 0】

VVCv1で定義された共通制約フラグ以外に、構文要素`gci_num_reserved_bits`において、追加の共通制約フラグがさらに規定される。`gci_num_reserved_bits`は、8ビットの符号なし整数であり、その値は、共通制約構文構造において伝送される追加ビットの数を示す。VVC仕様では、これらの追加ビット（構文要素`gci_reserved_zero_bit[ i ]`と呼ばれる）は、ビットストリームから抽出され、破棄される。このような規定により、VVCv1のデコーダは、少なくともVVCのそれ以降のバージョンで生成されたビットストリームの高レベル構文部分と前方互換性があることができる。

30

【 0 0 4 1】

VVCバージョン2（VVCv2）における共通制約情報について
VVCバージョン2の現在のドラフト（「VVC動作範囲拡張（ドラフト5）」、ITU-T SG 16 WP 3及びISO/IEC JTC 1/SC 29の共同ビデオエキスパートチームのドキュメント、JVET-X2005）では、共通制約フラグによって制約するいくつかの追加符号化ツールが提案されている。VVCv1で`gci_num_reserved_bits`と呼ばれる8ビットフィールドを`gci_num_additional_bits`に名称変更することが提案されている。VVCv2で調整が提案されている構文は、次の通りである。

40

【 0 0 4 2】

【表 2】

general_constraints_info() {	記述子
gci_present_flag	u(1)
if(gci_present_flag) {	
/* 共通general */	
gci_intra_only_constraint_flag	u(1)
...	
~~gci_num_reserved_bits~~	~~u(8)~~
gci_num_additional_bits	u(8)
if(gci_num_additional_bits > 0) {	
gci_all_rap_pictures_constraint_flag	u(1)
gci_no_extended_precision_processing_constraint_flag	u(1)
gci_no_ts_residual_coding_rice_constraint_flag	u(1)
gci_no_rrc_rice_extension_constraint_flag	u(1)
gci_no_persistent_rice_adaptation_constraint_flag	u(1)
gci_no_reverse_last_sig_coeff_constraint_flag	u(1)
numAdditionalBitsUsed = 6	
} else	
numAdditionalBitsUsed = 0	
for(i = 0; i < gci_num_additional_bits - numAdditionalBitsUsed; i++)	
~~for(i = 0; i < gci_num_reserved_bits; i++)~~	
gci_reserved_zero_bit[i]	u(1)
}	
while(!byte_aligned())	
gci_alignment_zero_bit	f(1)
}	

10

20

30

40

【0043】

追加の共通制約フラグが、gci_all_rap_pictures_constraint_flag、gci_no_extended_precision_processing_constraint_flag、gci_no_ts_residual_coding_rice_constraint_flag、gci_no_rrc_rice_extension_constraint_flag、gci_no_persistent_rice_adaptation_constraint_flag、及びgci_no_reverse_last_sig_coeff_constraint_flagという合計6つのものである。gci_num_additional

50

l__bits 構文要素の解釈（「セマンティック」）は、次の通りである。

【0044】

gci__num__additional__bits は、共通制約情報構文構造内の gci__alignment__zero__bit 構文要素（存在する場合）以外の追加 GCI ビットの数規定する。本ドキュメントの当該バージョンに準拠するビットストリームでは、gci__num__additional__bits の値は、0 又は 1 に等しくなるべきである。1 より大きい gci__num__additional__bits の値は、将来の使用のために ITU-T | ISO / IEC に予約されている。本ドキュメントの当該バージョンでは、gci__num__additional__bits の値が 0 又は 1 に等しいことが要求されているが、本ドキュメントの当該バージョンに準拠するデコーダは、1 より大きい gci__num__additional__bits の値が構文に存在することを許可し、且つ gci__num__additional__bits が 1 より大きい場合は、すべての gci__reserved__zero__bit[i] 構文要素の値を無視すべきである。

10

【0045】

VVCv2 のビットストリームが共通制約情報を伝送する場合、6 つの VVCv2 共通制約フラグを伝送しない場合は、構文要素 gci__num__additional__bits を 0 に設定し、6 つの VVCv2 共通制約フラグを伝送する場合、gci__num__additional__bits を 1 に設定すべきであることが提案された。

【0046】

しかし、上記の提案された VVCv2 の共通制約構文の仕様は、VVCv1 のデコーダとの非互換性を引き起こす。具体的に、追加の共通制約フラグを伝送する時に、提案された VVCv2 構文は、gci__num__additional__bits を 1 に設定することによって該フラグを伝送する。VVCv2 のデコーダでは、共通制約情報を伝送する時に、ビットストリームから、gci__num__additional__bits 構文要素を 8 ビットの符号なし整数に復号する。gci__num__additional__bits が値 1 に復号される場合、ビットストリームから 6 つの追加ビットが復号される。これらの 6 ビットを、VVCv2 で制約され得る追加の復号ツールの共通制約フラグとして解釈する。

20

【0047】

VVCv1 のデコーダでは、同じ 8 ビットフィールドが、gci__num__reserved__bits として解釈される。ただし、当該構文要素の値が 1 に復号されると、ビットストリームから 1 つの追加ビットのみが復号される。当該ビットは破棄され、使用されない。その結果、VVCv2 のビットストリームを復号している VVCv1 のデコーダは、ビットストリームに VVCv1 仕様で認識されない 5 つの追加ビットが存在するため、同期できない問題に遭遇する可能性がある。

30

【0048】

上述したように、同期できないことは、異なるバージョンのビデオ符号化標準間での重大な非互換性の問題である。VVCv1 のデコーダは、VVCv2 のビットストリームのすべてのコンテンツを復号できない。これは、VVCv2 のビットストリームが、VVCv2 仕様で定義されているが、VVCv1 のデコーダにとって知らない符号化ツールを使用しているためである。しかし、VVCv1 のデコーダは、少なくとも VVCv2 のビットストリームの高レベル構文部分を復号できることが望ましい。高レベル構文を正常に復号することにより、ビデオデコーダは、共通制約情報だけでなく、プロファイルとティア情報（Profile and tier information）も決定することができる。このような情報は、ビットストリームを復号するために必要な能力のヒントをデコーダに提供する。例えば、共通制約情報は、どの符号化ツールがビットストリームに制約されるかに関する指示をデコーダに提供する。プロファイルとティア情報は、サポートする必要がある非圧縮ビデオデータスループット（例えば、ビデオデータレート、フレームレート、解像度など）に関する指示をデコーダに提供する。

40

50

【0049】

高レベル構文が正常に復号された場合、ビデオデコーダは、現在のビットストリームを復号できるか否かを決定することができ、復号できない場合、デコーダは、復号プロセスを正常に終了することができる。これに対し、高レベル構文の復号に同期できないことは、高レベル構文で提供される情報が正しく復号できないことを意味する。最悪の場合、デコーダは、同期できないことが発生した後に、完全に誤った構文要素の値として復号する可能性があり、その結果、パラメータ設定の誤りや、後続の低レベル構文の復号の誤りにもつながり、復号が失敗する可能性がある。

【0050】

さらに、VVCv2のデコーダでは、共通制約情報を伝送する時に、ビットストリームから、`gci_num_additional_bits` 構文要素を、8ビットの符号なし整数として復号する。`gci_num_additional_bits`の値が0に復号された場合、それ以上の共通制約フラグは伝送されない。この場合、追加の共通制約フラグに対して推測値が規定されず、追加の共通制約フラグの値は未定義である。したがって、追加の共通制約フラグに関連する符号化ツールが制約されるべきなのか、それとも制約されるべきでないのかは不明確であり、デコーダの実現に整合性がなくなる可能性がある。

10

【0051】

VVC仕様では、構文要素 `gci_reserved_zero_bit[i]` の名称は、そのような構文要素の値が0でなければならないと誤解を招くように示唆している。通常、エンコーダは、予約された構文要素をプレースホルダとしてビットストリームに書き込む場合、予約された構文要素の名前にデフォルト値が埋め込まれる。しかし、共通制約構文構造の設計は、エンコーダが `gci_reserved_zero_bit[i]` を絶対書き込むことにならない。`gci_reserved_zero_bit[i]` は、特定のバージョンのVVCデコーダがより高いバージョンのVVCビットストリームを読み取る場合にのみ使用される。この場合、`gci_reserved_zero_bit[i]` の値が0であることは保証されない。以下では、上記の問題を解決するためのいくつかの解決策を提案する。

20

【0052】

共通制約情報フラグの伝送について

30

1つの実施例では、共通制約情報を伝送することによって上記の同期できない問題を解決する。VVCv2のビットストリームが共通制約情報を伝送し（即ち、`gci_present_flag`の値が1である）、且つVVCv2共通制約情報がN個の制約される可能性のある追加符号化ツールで構成される場合、構文要素 `gci_num_additional_bits` の値は、0又はNのみに設定できる。`gci_num_additional_bits` が0に設定されている場合、これらのN個の追加符号化ツールの共通制約フラグは伝送されない。`gci_num_additional_bits` がNに設定されている場合、ビットストリームの後続のNビットは、これらのN個の追加符号化ツールの共通制約フラグを伝送するために使用される。1つの例では、Nは6に設定される。

40

【0053】

VVCv2のビットストリームでは、`gci_num_additional_bits` を0又はN以外の値に設定することが許可されない。ただし、VVCv2のデコーダは、`gci_num_additional_bits` を0又はN以外の値に設定する共通制約情報を処理できる。例えば、`gci_num_additional_bits` は、0より大きく、且つNより小さい値、又はNより大きい値に設定できる。`gci_num_additional_bits` の復号された値をMとする。すると、Mが0より大きく、且つNより小さい（即ち、 $0 < M < N$ ）場合、`gci_num_additional_bits` 構文要素を復号した後、デコーダは、ビットストリームからMビットを `gci_reserved_zero_bit[i]` 構文要素として抽出して破棄する。Mが

50

Nより大きい（即ち、 $N < M$ ）場合、`gci_num_additional_bits` 構文要素を復号した後、デコーダは、ビットストリームからNビットを抽出し、これらのNビットを、N個の追加符号化ツールの共通制約フラグとして解釈する。次に、デコーダは、ビットストリームから $(M - N)$ ビットを`gci_reserved_zero_bit[i]` 構文要素としてさらに抽出して破棄する。

【0054】

他の例では、`VVCv2`のデコーダは、`gci_num_additional_bits`を0より大きく、且つNより小さい値に設定する共通制約情報処理する必要はない。正当な`VVCv1`ビットストリームは、`gci_num_reserved_bits`の値を0のみに設定できる。正当な`VVCv2`のビットストリームは、`gci_num_additional_bits`の値を0又はNのみに設定できる。未来バージョンの`VVC`のビットストリームでは、`gci_num_additional_bits`を0からNの間の値に設定することは許可されない。

10

【0055】

本実施例の1つの例では、`VVCv2`の共通制約情報構文を、`VVCv2`の符号化ツールのために現在提案されている6つの共通制約フラグに修正し、当該修正は以下の表1に示すとおりであり（追加の内容は下線付きで示され、削除された内容を取り消し線付きで示される（訳注：明細書本文の取り消し線が引かれた部分を《》内に記載している）、ここで、「`if (gci_num_additional_bits > 0)`」は「`if (gci_num_additional_bits > 5)`」に置き換えられている。

20

【0056】

30

40

50

【表 3】

general_constraints_info() {	記述子
gci_present_flag	u(1)
if(gci_present_flag) {	
/*共通general*/	
gci_intra_only_constraint_flag	u(1)
...	
gci_num_additional_bits	u(8)
if(gci_num_additional_bits > 0.5) {	
gci_all_rap_pictures_constraint_flag	u(1)
gci_no_extended_precision_processing_constraint_flag	u(1)
gci_no_ts_residual_coding_rice_constraint_flag	u(1)
gci_no_rrc_rice_extension_constraint_flag	u(1)
gci_no_persistent_rice_adaptation_constraint_flag	u(1)
gci_no_reverse_last_sig_coeff_constraint_flag	u(1)
numAdditionalBitsUsed = 6	
} else	
numAdditionalBitsUsed = 0	
for(i = 0; i < gci_num_additional_bits - numAdditionalBitsUsed; i++)	
gci_reserved_zero_bit[i]	u(1)
}	
while(!byte_aligned())	
gci_alignment_zero_bit	f(1)
}	

10

20

30

【0057】

1つの例では、VVCv2の符号化ツールに6つの追加の共通制約フラグが存在する場合、gci_num_additional_bitsの対応するセマンティックは、次の通りである（追加の内容は下線付きで示され、削除された内容を取り消し線付きで示される（訳注：明細書本文の取り消し線が引かれた部分を《》内に記載している））。

【0058】

gci_num_additional_bitsは、共通制約情報構文構造内のgci_alignment_zero_bit構文要素（存在する場合）以外の追加GCIビットの数を規定する。本ドキュメントの当該バージョンに準拠するビットストリームでは、gci_num_additional_bitsの値は、0又は《1》6に等しくなるべきである。gci_num_additional_bitsの《1より大きい》0及び6以外の値は、ITU-T | ISO / IECに予約されて将来の使用に用いられる。本ドキュメントの当該バージョンでは、gci_num_additional_bitsの値が0又は《1》6に等しいことが要求されているが、本ドキュメントの当該バージョンに準拠するデコーダは、《1より大きい》0及び6以外のgci_num_additional_bitsの値が構文に存在することを許可し、且つgci_num_additional_bitsが《1より大きい》0及び6以外である場合、すべてのg

40

50

`g c i _ r e s e r v e d _ z e r o _ b i t [ i ]` 構文要素の値を無視すべきである。
【0059】

`g c i _ n u m _ a d d i t i o n a l _ b i t s` のセマンティックの上記の例では、上記の0又は6の値に加えて、`g c i _ n u m _ a d d i t i o n a l _ b i t s` は、0又は6以外の値を取ることが許可される。つまり、`g c i _ n u m _ a d d i t i o n a l _ b i t s` の値は、1～5の間であっても良い。`g c i _ n u m _ a d d i t i o n a l _ b i t s` が6より大きい値を取ることにも許可される。`g c i _ n u m _ a d d i t i o n a l _ b i t s` の値が1～5（Mで示される）の間である場合、上記の表1に示す構文に従って、デコーダは、「if」条件がtrueであるときに実行されるステップをスキップし、「else」ステップに遷移して、「numAdditionalBitsUsed」の値を0に割り当てる。次に、「for」ループでMビットを読み取って破棄する。このようにして、同期できない問題を回避することができる。`g c i _ n u m _ a d d i t i o n a l _ b i t s` の値Mが6より大きい場合、6つの追加の共通制約フラグを抽出し、残りのM-6ビットをさらに抽出して破棄する。

10

【0060】

別の例では、VVCv2の符号化ツールに6つの追加の共通制約フラグが存在する場合、`g c i _ n u m _ a d d i t i o n a l _ b i t s` の対応するセマンティックは、次の通りである（追加の内容は下線付きで示され、削除された内容は取り消し線付きで示される（訳注：明細書本文の取り消し線が引かれた部分を《》内に記載している））。

【0061】

`g c i _ n u m _ a d d i t i o n a l _ b i t s` は、共通制約情報構文構造内の `g c i _ a l i g n m e n t _ z e r o _ b i t` 構文要素（存在する場合）以外の追加GCIビットの数を規定する。本ドキュメントの当該バージョンに準拠するビットストリームでは、`g c i _ n u m _ a d d i t i o n a l _ b i t s` の値は、0又は《1》6に等しくなるべきである。`g c i _ n u m _ a d d i t i o n a l _ b i t s` の《1》6より大きい値は、ITU-T | ISO / IECに予約されて将来の使用のために用いられる。本ドキュメントの当該バージョンでは、`g c i _ n u m _ a d d i t i o n a l _ b i t s` の値が0又は《1》6に等しいことが要求されているが、本ドキュメントの当該バージョンに準拠するデコーダは、《1》6より大きい `g c i _ n u m _ a d d i t i o n a l _ b i t s` の値が構文に存在することを許可し、且つ `g c i _ n u m _ a d d i t i o n a l _ b i t s` が《1》6より大きい場合、すべての `g c i _ r e s e r v e d _ z e r o _ b i t [ i ]` 構文要素の値を無視すべきである。

20

30

【0062】

共通制約情報を伝送する別の実施例では、VVCv2のビットストリームが共通制約情報を伝送し（即ち、`g c i _ p r e s e n t _ f l a g` の値が1である）、且つVVCv2の共通制約情報がN個の制約される可能性のある追加符号化ツールで構成される場合、構文要素 `g c i _ n u m _ a d d i t i o n a l _ b i t s` の値をMに設定できる。Mの範囲は、0からN（0及びNを含む）（ $0 \leq M \leq N$ ）である。`g c i _ n u m _ a d d i t i o n a l _ b i t s` が0に設定されている場合、これらのN個の追加符号化ツールの共通制約フラグは伝送されない。`g c i _ n u m _ a d d i t i o n a l _ b i t s` が非ゼロ値のMに設定されている場合、ビットストリームの後続のMビットは、これらのN個の追加符号化ツールのうちのM個の共通制約フラグを伝送するために使用される。どのM個の追加符号化ツールが制約されるかは、共通制約フラグが共通制約情報構文テーブルに出現する順序によって決定される。

40

【0063】

当該実施例の1つの例では、VVCv2の共通制約情報構文を符号化ツールのために現在提案されている6つの共通制約フラグに修正し、該修正は次の通りであり得る。

【0064】

50

【表 4】

general_constraints_info() {	記述子
gci_present_flag	u(1)
if(gci_present_flag) {	
/* 共通general */	
gci_intra_only_constraint_flag	u(1)
...	
gci_num_additional_bits	u(8)
if(gci_num_additional_bits > 0)	
gci_all_rap_pictures_constraint_flag	u(1)
if(gci_num_additional_bits > 1)	
gci_no_extended_precision_processing_constraint_flag	u(1)
if(gci_num_additional_bits > 2)	
gci_no_ts_residual_coding_rice_constraint_flag	u(1)
if(gci_num_additional_bits > 3)	
gci_no_rrc_rice_extension_constraint_flag	u(1)
if(gci_num_additional_bits > 4)	
gci_no_persistent_rice_adaptation_constraint_flag	u(1)
if(gci_num_additional_bits > 5)	
gci_no_reverse_last_sig_coeff_constraint_flag	u(1)
if(gci_num_additional_bits > 6) {	
numAdditionalBitsUsed = 6	
for(i = 0; i < gci_num_additional_bits - numAdditionalBitsUsed; i++)	
gci_reserved_zero_bit[i]	u(1)
}	
}	
while(!byte_aligned())	
gci_alignment_zero_bit	f(1)
}	

10

20

30

40

【 0 0 6 5 】

当該実施例の別の例では、よりコンパクトな構文テーブルによって同等の動作を実現することができる。

【 0 0 6 6 】

【表 5】

general_constraints_info() {	記述子
gci_present_flag	u(1)
if(gci_present_flag) {	
/* 共通general */	
gci_intra_only_constraint_flag	u(1)
...	
gci_num_additional_bits	u(8)
if(gci_num_additional_bits > 0)	
gci_all_rap_pictures_constraint_flag	u(1)
if(gci_num_additional_bits > 1)	
gci_no_extended_precision_processing_constraint_flag	u(1)
if(gci_num_additional_bits > 2)	
gci_no_ts_residual_coding_rice_constraint_flag	u(1)
if(gci_num_additional_bits > 3)	
gci_no_rrc_rice_extension_constraint_flag	u(1)
if(gci_num_additional_bits > 4)	
gci_no_persistent_rice_adaptation_constraint_flag	u(1)
if(gci_num_additional_bits > 5)	
gci_no_reverse_last_sig_coeff_constraint_flag	u(1)
for(i = 0; i < gci_num_additional_bits-6; i++)	
gci_reserved_zero_bit[i]	u(1)
}	
}	
while(!byte_aligned())	
gci_alignment_zero_bit	f(1)
}	

10

20

30

【0067】

VVCv2の符号化ツールの共通制約フラグの順序を変更することによって、共通制約情報構文で当該実施例の代替配置を表現することができる。

40

【0068】

VVCv2の符号化ツールが6つの追加の共通制約フラグを有する場合、gci_num_additional_bitsの対応するセマンティックは、次の通りであり得る。

【0069】

gci_num_additional_bitsは、共通制約情報構文構造内のgci_alignment_zero_bit構文要素（存在する場合）以外の追加GCIビットの数を規定する。本ドキュメントの当該バージョンに準拠するビットストリームでは、gci_num_additional_bitsの値は、0～6の範囲（0及び6を含む）内であるべきである。gci_num_additional_bitsの6よ

50

り大きい値は、ITU-T | ISO / IEC に予約されて将来の使用のために用いられる。本ドキュメントの当該バージョンでは、`gci_num_additional_bits` の値が 0 ～ 6 の範囲（0 及び 6 を含む）内であることが要求されているが、本ドキュメントの当該バージョンに準拠するデコーダは、6 より大きい `gci_num_additional_bits` の値が構文に存在することを許可し、且つ `gci_num_additional_bits` が 6 より大きい場合、すべての `gci_reserved_zero_bit[i]` 構文要素の値を無視すべきである。

【0070】

本実施例では、`gci_num_additional_bits` の値に基づいて、追加の共通制約フラグの一部又はすべてが伝送されない場合がある。本実施例の 1 つの配置では、VVCv2 のビットストリームが共通制約情報を伝送する場合（即ち、`gci_present_flag` の値が 1 である場合）、伝送されていない追加の共通制約フラグに対応する符号化ツールに制約をかけない。下記のように、この振舞いは、`gci_num_additional_bits` のセマンティックを変更することによって表すことができる。

【0071】

`gci_num_additional_bits` は、共通制約情報構文構造内の `gci_alignment_zero_bit` 構文要素（存在する場合）以外の追加 GCI ビットの数に規定する。本ドキュメントの当該バージョンに準拠するビットストリームでは、`gci_num_additional_bits` の値の範囲は、0 ～ 6 の（0 及び 6 を含む）内であるべきである。`gci_num_additional_bits` の 6 より大きい値は、ITU-T | ISO / IEC に予約されて将来の使用のために用いられる。本ドキュメントの当該バージョンでは、`gci_num_additional_bits` の値が 0 ～ 6 の範囲（0 及び 6 を含む）内であることが要求されているが、本ドキュメントの当該バージョンに準拠するデコーダは、6 より大きい `gci_num_additional_bits` の値が構文に存在することを許可し、且つ `gci_num_additional_bits` が 6 より大きい場合、すべての `gci_reserved_zero_bit[i]` 構文要素の値を無視すべきである。さらに、`gci_present_flag` が 1 に等しい場合、`general_constraints_info()` の構文内に対応する制約フラグが存在しない符号化ツールには制約をかけない。

【0072】

本実施例の別の 1 つの配置では、VVCv2 のビットストリームが共通制約情報を伝送し（即ち、`gci_present_flag` の値が 1 である場合）、且つ追加の共通制約フラグが伝送されていない場合、対応するツールが制約される。下記のように、この振舞いは、追加の共通制約フラグのセマンティックを変更することによって表すことができる。

【0073】

`gci_num_additional_bits` は、共通制約情報構文構造内の `gci_alignment_zero_bit` 構文要素（存在する場合）以外の追加 GCI ビットの数に規定する。本ドキュメントの当該バージョンに準拠するビットストリームでは、`gci_num_additional_bits` の値は、0 ～ 6 の範囲（0 及び 6 を含む）内であるべきである。`gci_num_additional_bits` の 6 より大きい値は、ITU-T | ISO / IEC に予約されて将来の使用に用いられる。本ドキュメントの当該バージョンでは、`gci_num_additional_bits` の値の範囲は 0 ～ 6 の（0 と 6 を含む）であることが要求されているが、本ドキュメントの当該バージョンに準拠するデコーダは、6 より大きい `gci_num_additional_bits` の値が構文に存在することを許可し、且つ `gci_num_additional_bits` が 6 より大きい場合、すべての `gci_reserved_zero_bit[i]` 構文要素の値を無視すべきである。

【0074】

gci_all_rap_pictures_constraint_flagが1に等しい場合、OlsInScope内のすべてのピクチャがph_recovery_poc_cntが0に等しいGDRピクチャ、又はIRAPピクチャであることを規定する。gci_all_rap_pictures_constraint_flagが0に等しい場合、このような制約をかけない。gci_present_flagが1に等しく、且つgci_all_rap_pictures_constraint_flagが存在しない場合、gci_all_rap_pictures_constraint_flagの値が1に等しいと推測される。

【0075】

gci_no_extended_precision_processing_constraint_flagが1に等しい場合、OlsInScope内のすべてのピクチャのsps_extended_precision_flagがいずれも0に等しくなるべきであることを規定する。gci_no_extended_precision_processing_constraint_flagが0に等しい場合、このような制約をかけない。gci_present_flagが1に等しく、且つgci_no_extended_precision_processing_constraint_flagが存在しない場合、gci_no_extended_precision_processing_constraint_flagの値が1に等しいと推測される。

【0076】

gci_no_ts_residual_coding_rice_constraint_flagが1に等しい場合、OlsInScope内のすべてのピクチャのsps_ts_residual_coding_rice_present_in_sh_flagが0に等しくなるべきであることを規定する。gci_no_ts_residual_coding_rice_constraint_flagが0に等しい場合、このような制約をかけない。gci_present_flagが1に等しく、且つgci_no_ts_residual_coding_rice_constraint_flagが存在しない場合、gci_no_ts_residual_coding_rice_constraint_flagの値が1に等しいと推測される。

【0077】

gci_no_rrc_rice_extension_constraint_flagが1に等しい場合、OlsInScope内のすべてのピクチャのsps_rrc_rice_extension_flagが0に等しくなるべきであることを規定する。gci_no_rrc_rice_extension_constraint_flagが0に等しい場合、このような制約をかけない。gci_present_flagが1に等しく、且つgci_no_rrc_rice_extension_constraint_flagが存在しない場合、gci_no_rrc_rice_extension_constraint_flagの値が1に等しいと推測される。

【0078】

gci_no_persistent_rice_adaptation_constraint_flagが1に等しい場合、OlsInScope内のすべてのピクチャのsps_persistent_rice_adaptation_enabled_flagが0に等しくなるべきであることを規定する。gci_no_persistent_rice_adaptation_constraint_flagが0に等しい場合、このような制約をかけない。gci_present_flagが1に等しく、且つgci_no_persistent_rice_adaptation_constraint_flagが存在しない場合、gci_no_persistent_rice_adaptation_constraint_flagの値が1に等しいと推測される。

【0079】

gci_no_reverse_last_sig_coeff_constraint_flagが1に等しい場合、OlsInScope内のすべてのピクチャのsps_reverse_last_sig_coeff_enabled_flagが0に等しくなるべきであることを規定する。gci_no_reverse_last_sig_coeff_constraint_flagが0に等しい場合、このような制約をかけない。gci_present_flagが1に等しく、且つgci_no_reverse_last_sig_coeff_constraint_flagが存在しない場合、gci_no_reverse_last_sig_coeff_constraint_flagの値が1に等しいと推測される。

【0080】

共通制約情報フラグの初期化について

上述のデコーダ実現の不明確と不整合の問題を解決するために、共通制約情報フラグを初期化する1つの実施例について説明する。当該実施例では、gci_present_flagが1に等しく、且つgci_num_additional_bitsが0に等しい場合、general_constraints_info()は、gci_all_rap_pictures_constraint_flag、gci_no_extended_precision_processing_constraint_flag、gci_no_ts_residual_coding_rice_constraint_flag、gci_no_reverse_last_sig_coeff_constraint_flag、gci_no_rrc_rice_extension_constraint_flag、及びgci_no_persistent_rice_adaptation_constraint_flagに関連する符号化ツールに制約をかけない。

【0081】

一例として、gci_num_additional_bitsのセマンティックの可能な変化は、次に示す通りであり、これらの変化は、VVCバージョン2に定められた現在バージョンに基づくものである（追加の内容は下線付きで示され、削除された内容は取り消し線付きで示される（訳注：明細書本文の取り消し線が引かれた部分を《》内に記載している））。

【0082】

gci_num_additional_bitsは、共通制約情報構文構造内のgci_alignment_zero_bit構文要素（存在する場合）以外の追加GCIビットの数を規定する。本ドキュメントの当該バージョンに準拠するビットストリームでは、gci_num_additional_bitsの値は、0又は1に等しくなるべきである。gci_num_additional_bitsの1より大きい値は、ITU-T | ISO / IECに予約されて将来の使用に用いられる。本ドキュメントの当該バージョンでは、gci_num_additional_bitsの値が0又は1に等しいことが要求されているが、本ドキュメントの当該バージョンに準拠するデコーダは、1より大きいgci_num_additional_bitsの値が構文に存在することを許可し、且つgci_num_additional_bitsが1より大きい場合、すべてのgci_reserved_zero_bit[i]構文要素の値を無視すべきである。gci_present_flagが1に等しく、且つgci_num_additional_bitsが0に等しい場合、general_constraints_info()は、gci_all_rap_pictures_constraint_flag、gci_no_extended_precision_processing_constraint_flag、gci_no_ts_residual_coding_rice_constraint_flag、gci_no_reverse_last_sig_coeff_constraint_flag、gci_no_rrc_rice_extension_constraint_flag、及びgci_no_persistent_rice_adaptation_constraint

10

20

30

40

50

flagに関連する符号化ツールに制約をかけない。

【0083】

別の例として、`gci_num_additional_bits`のセマンティックの可能な変化は、次に示す通りであり、これらの変化は、VVCバージョン2に定められた現在バージョンに基づくものである。

【0084】

`gci_num_additional_bits`は、共通制約情報構文構造内の`gci_alignment_zero_bit`構文要素（存在する場合）以外の追加GCIビットの数を規定する。本ドキュメントの当該バージョンに準拠するビットストリームでは、`gci_num_additional_bits`の値は、0又は $\ll 1 \gg 6$ に等しくなるべきである。`gci_num_additional_bits`の $\ll 1$ より大きい $\gg 0$ 及び6以外の値は、ITU-T | ISO / IECに予約されて将来の使用に用いられる。本ドキュメントの当該バージョンでは、`gci_num_additional_bits`の値が0又は $\ll 1 \gg 6$ に等しいことが要求されているが、本ドキュメントの当該バージョンに準拠するデコーダは、 $\ll 1$ より大きい $\gg 0$ 及び6以外の`gci_num_additional_bits`の値が構文に存在することを許可し、且つ`gci_num_additional_bits`が $\ll 1$ より大きい $\gg 0$ 及び6以外である場合、すべての`gci_reserved_zero_bit[i]`構文要素の値を無視すべきである。`gci_present_flag`が1に等しく、`gci_num_additional_bits`が0に等しい場合、`general_constraints_info()`は、`gci_all_rap_pictures_constraint_flag`、`gci_no_extended_precision_processing_constraint_flag`、`gci_no_ts_residual_coding_rice_constraint_flag`、`gci_no_reverse_last_sig_coeff_constraint_flag`、`gci_no_rrc_rice_extension_constraint_flag`、及び`gci_no_persistent_rice_adaptation_constraint_flag`に関連する符号化ツールに制約をかけない。

【0085】

VVCバージョン2に定められた現在バージョンでは、`gci_num_additional_bits`のセマンティックが`gci_present_flag`が1に等しいときにしか使用されない。`gci_present_flag`が0に等しいシナリオは、VVCバージョン2仕様の別のセクションで扱われる。したがって、上記の`gci_num_additional_bits`のセマンティックの「`gci_present_flag`が1に等しい場合」が自動的に満たされる。さらに、VVCバージョン2仕様では、`gci_num_additional_bits`の値が1～5を取ることは許可されないため、「`gci_num_additional_bits`が0に等しい」は、「`gci_num_additional_bits`が5以下である」、又は「`gci_all_rap_pictures_constraint_flag`、`gci_no_extended_precision_processing_constraint_flag`、`gci_no_ts_residual_coding_rice_constraint_flag`、`gci_no_reverse_last_sig_coeff_constraint_flag`、`gci_no_rrc_rice_extension_constraint_flag`、及び`gci_no_persistent_rice_adaptation_constraint_flag`が存在しない」と同等になる。したがって、`gci_num_additional_bits`のセマンティックに対する上記の変更は、次の内容と同等である。

【0086】

`gci_num_additional_bits`は、共通制約情報構文構造内の`gci_alignment_zero_bit`構文要素（存在する場合）以外の追加GCI

ビットの数を規定する。本ドキュメントの当該バージョンに準拠するビットストリームでは、`gci_num_additional_bits`の値は、0又は≪1≫6に等しくなる。`gci_num_additional_bits`の≪1より大きい≫0及び6以外の値は、ITU-T | ISO / IECに予約されて将来の使用に用いられる。本ドキュメントの当該バージョンでは、`gci_num_additional_bits`の値が0又は≪1≫6に等しいことが要求されているが、本ドキュメントの当該バージョンに準拠するデコーダは、≪1より大きい≫0及び6以外の`gci_num_additional_bits`の値が構文に存在することを許可し、且つ`gci_num_additional_bits`が≪1より大きい≫0及び6以外である場合、すべての`gci_reserved_zero_bit[i]`構文要素の値を無視すべきである。`gci_all_rap_pictures_constraint_flag`、`gci_no_extended_precision_processing_constraint_flag`、`gci_no_ts_residual_coding_rice_constraint_flag`、`gci_no_reverse_last_sig_coeff_constraint_flag`、`gci_no_rrc_rice_extension_constraint_flag`、及び`gci_no_persistent_rice_adaptation_constraint_flag`が存在しない場合、`general_constraints_info()`は、`gci_all_rap_pictures_constraint_flag`、`gci_no_extended_precision_processing_constraint_flag`、`gci_no_ts_residual_coding_rice_constraint_flag`、`gci_no_reverse_last_sig_coeff_constraint_flag`、`gci_no_rrc_rice_extension_constraint_flag`、及び`gci_no_persistent_rice_adaptation_constraint_flag`に関連する符号化ツールに制約をかけない。

10

20

30

40

50

【0087】

以下はいくつかの例であり、ここで、上記のセマンティックと同様の方式で追加の共通制約フラグの推測値を設定できる。例えば、追加の共通制約フラグのセマンティックは、推測値の設定を含めるように次のように変更できる。

【0088】

`gci_all_rap_pictures_constraint_flag`が1に等しい場合、`OlsInScope`内のすべてのピクチャが`ph_recovery_poc_cnt`が0に等しいGDRピクチャ、又はIRAPピクチャであることを規定する。`gci_all_rap_pictures_constraint_flag`が0に等しい場合、このような制約をかけない。存在しない場合、`gci_all_rap_pictures_constraint_flag`の値が0に等しいと推測される。

【0089】

`gci_no_extended_precision_processing_constraint_flag`が1に等しい場合、`OlsInScope`内のすべてのピクチャの`sps_extended_precision_flag`が0に等しくなるべきであることを規定する。`gci_no_extended_precision_processing_constraint_flag`が0に等しい場合、このような制約をかけない。存在しない場合、`gci_no_extended_precision_processing_constraint_flag`の値が0に等しいと推測される。

【0090】

`gci_no_ts_residual_coding_rice_constraint_flag`が1に等しい場合、`OlsInScope`内のすべてのピクチャの`sps_ts_residual_coding_rice_present_in_sh_flag`が0に等しくなるべきであることを規定する。`gci_no_ts_residu`

al_coding_rice_constraint_flagが0に等しい場合、このような制約をかけない。存在しない場合、gci_no_ts_residual_coding_rice_constraint_flagの値が0に等しいと推測される。

【0091】

gci_no_rrc_rice_extension_constraint_flagが1に等しい場合、OlsInScope内のすべてのピクチャのsps_rrc_rice_extension_flagが0に等しくなるべきであることを規定する。gci_no_rrc_rice_extension_constraint_flagが0に等しい場合、このような制約をかけない。存在しない場合、gci_no_rrc_rice_extension_constraint_flagの値が0に等しいと推測される。

10

【0092】

gci_no_persistent_rice_adaptation_constraint_flagが1に等しい場合、OlsInScope内のすべてのピクチャのsps_persistent_rice_adaptation_enabled_flagが0に等しくなるべきであることを規定する。gci_no_persistent_rice_adaptation_constraint_flagが0に等しい場合、このような制約をかけない。存在しない場合、gci_no_persistent_rice_adaptation_constraint_flagの値が0に等しいと推測される。

20

【0093】

gci_no_reverse_last_sig_coeff_constraint_flagが1に等しい場合、OlsInScope内のすべてのピクチャのsps_reverse_last_sig_coeff_enabled_flagが0に等しくなるべきであることを規定する。gci_no_reverse_last_sig_coeff_constraint_flagが0に等しい場合、このような制約をかけない。存在しない場合、gci_no_reverse_last_sig_coeff_constraint_flagの値が0に等しいと推測される。

【0094】

別の例では、追加の共通制約フラグのセマンティックは、次のように変更される。

30

【0095】

gci_all_rap_pictures_constraint_flagが1に等しい場合、OlsInScope内のすべてのピクチャがph_recovery_poc_cntが0に等しいGDRピクチャ、又はIRAPピクチャであることを規定する。gci_all_rap_pictures_constraint_flagが0に等しい場合、このような制約をかけない。gci_all_rap_pictures_constraint_flagが存在しない場合、その値が0に等しいと推測される。

【0096】

gci_no_extended_precision_processing_constraint_flagが1に等しい場合、OlsInScope内のすべてのピクチャのsps_extended_precision_flagが0に等しくなるべきであることを規定する。gci_no_extended_precision_processing_constraint_flagが0に等しい場合、このような制約をかけない。gci_no_extended_precision_processing_constraint_flagが存在しない場合、その値が0に等しいと推測される。

40

【0097】

gci_no_ts_residual_coding_rice_constraint_flagが1に等しい場合、OlsInScope内のすべてのピクチャのsps

50

gci_no_ts_residual_coding_rice_present_in_sh_flagが0に等しくなるべきであることを規定する。gci_no_ts_residual_coding_rice_constraint_flagが0に等しい場合、このような制約をかけない。gci_no_ts_residual_coding_rice_constraint_flagが存在しない場合、その値が0に等しいと推測される。

【0098】

gci_no_rrc_rice_extension_constraint_flagが1に等しい場合、OlsInScope内のすべてのピクチャのsps_rrc_rice_extension_flagが0に等しくなるべきであることを規定する。gci_no_rrc_rice_extension_constraint_flagが0に等しい場合、このような制約をかけない。gci_no_rrc_rice_extension_constraint_flagが存在しない場合、その値が0に等しいと推測される。

10

【0099】

gci_no_persistent_rice_adaptation_constraint_flagが1に等しい場合、OlsInScope内のすべてのピクチャのsps_persistent_rice_adaptation_enabled_flagが0に等しくなるべきであることを規定する。gci_no_persistent_rice_adaptation_constraint_flagが0に等しい場合、このような制約をかけない。gci_no_persistent_rice_adaptation_constraint_flagが存在しない場合、その値が0に等しいと推測される。

20

【0100】

gci_no_reverse_last_sig_coeff_constraint_flagが1に等しい場合、OlsInScope内のすべてのピクチャのsps_reverse_last_sig_coeff_enabled_flagが0に等しくなるべきであることを規定する。gci_no_reverse_last_sig_coeff_constraint_flagが0に等しい場合、このような制約をかけない。gci_no_reverse_last_sig_coeff_constraint_flagが存在しない場合、その値が0に等しいと推測される。

30

【0101】

別の例では、gci_num_additional_bitsのセマンティックは、次のように変更される。

【0102】

gci_num_additional_bitsは、共通制約情報構文構造内のgci_alignment_zero_bit構文要素（存在する場合）以外の追加GCIビットの数を規定する。本ドキュメントの当該バージョンに準拠するビットストリームでは、gci_num_additional_bitsの値は、0又は $\ll 1 \gg 6$ に等しくなるべきである。gci_num_additional_bitsの $\ll 1 \gg 6$ より大きい値は、ITU-T | ISO / IECに予約されて将来の使用に用いられる。本ドキュメントの当該バージョンでは、gci_num_additional_bitsの値が0又は $\ll 1 \gg 6$ に等しいことが要求されているが、本ドキュメントの当該バージョンに準拠するデコーダは、 $\ll 1 \gg 6$ より大きいgci_num_additional_bitsの値が構文に存在することを許可し、且つgci_num_additional_bitsが $\ll 1 \gg 6$ より大きい場合、すべてのgci_reserved_zero_bit[i]構文要素の値を無視すべきである。gci_num_additional_bitsが0に等しい場合、追加GCIビットによって規定されたすべての制約フラグが0に等しいと推測する。

40

【0103】

50

別の例では、`g c i _ n u m _ a d d i t i o n a l _ b i t s`のセマンティックは、次のように変更される。

【0104】

`g c i _ n u m _ a d d i t i o n a l _ b i t s`は、共通制約情報構文構造内の `g c i _ a l i g n m e n t _ z e r o _ b i t` 構文要素（存在する場合）以外の追加GCIビットの数を規定する。本ドキュメントの当該バージョンに準拠するビットストリームでは、`g c i _ n u m _ a d d i t i o n a l _ b i t s`の値は、0又は $\ll 1 \gg 6$ に等しくなるべきである。`g c i _ n u m _ a d d i t i o n a l _ b i t s`の $\ll 1 \gg 6$ より大きい値は、ITU-T | ISO / IECに予約されて将来の使用に用いられる。本ドキュメントの当該バージョンでは、`g c i _ n u m _ a d d i t i o n a l _ b i t s`の値が0又は $\ll 1 \gg 6$ に等しいことが要求されているが、本ドキュメントの当該バージョンに準拠するデコーダは、 $\ll 1 \gg 6$ より大きい`g c i _ n u m _ a d d i t i o n a l _ b i t s`の値が構文に存在することを許可し、且つ`g c i _ n u m _ a d d i t i o n a l _ b i t s`が $\ll 1 \gg 6$ より大きい場合、すべての`g c i _ r e s e r v e d _ z e r o _ b i t [ i ]` 構文要素の値を無視すべきである。存在しない場合、追加GCIビットによって規定されたすべての制約フラグが0に等しいと推測する。

10

【0105】

別の例では、共通制約情報構文要素の構文とセマンティックは、次のように変更される。

【0106】

20

30

40

50

【表 6】

general_constraints_info() {	記述子
gci_present_flag	u(1)
if(gci_present_flag) {	
/* 共通general */	
gci_intra_only_constraint_flag	u(1)
...	
gci_num_additional_bits	u(8)
if(gci_num_additional_bits > 0) {	
additional_general_constraints_info()	
gei_all_rap_pictures_constraint_flag	u(1)
gei_no_extended_precision_processing_constraint_flag	u(1)
gei_no_ts_residual_coding_rice_constraint_flag	u(1)
gei_no_rrc_rice_extension_constraint_flag	u(1)
gei_no_persistent_rice_adaptation_constraint_flag	u(1)
gei_no_reverse_last_sig_coeff_constraint_flag	u(1)
numAdditionalBitsUsed = 6	
} else	
numAdditionalBitsUsed = 0	
for(i = 0; i < gci_num_additional_bits - numAdditionalBitsUsed; i++)	
gci_reserved_zero_bit[i]	u(1)
}	
while(!byte_aligned())	
gci_alignment_zero_bit	f(1)
}	

【 0 1 0 7 】

10

20

30

40

50

【表 7】

<u>additional_general_constraints_info()</u> {	記述子
<u>gci_all_rap_pictures_constraint_flag</u>	<u>u(1)</u>
<u>gci_no_extended_precision_processing_constraint_flag</u>	<u>u(1)</u>
<u>gci_no_ts_residual_coding_rice_constraint_flag</u>	<u>u(1)</u>
<u>gci_no_rrc_rice_extension_constraint_flag</u>	<u>u(1)</u>
<u>gci_no_persistent_rice_adaptation_constraint_flag</u>	<u>u(1)</u>
<u>gci_no_reverse_last_sig_coeff_constraint_flag</u>	<u>u(1)</u>
}	

10

【0108】

gci_num_additional_bitsは、共通制約情報構文構造内の gci_alignment_zero_bit 構文要素（存在する場合）以外の追加 GCI ビットの数に規定する。本ドキュメントの当該バージョンに準拠するビットストリームでは、gci_num_additional_bitsの値は、0又は<<1>>6に等しくなるべきである。gci_num_additional_bitsの<<1>>6より大きい値は、ITU-T | ISO / IECに予約されて将来の使用に用いられる。本ドキュメントの当該バージョンでは、gci_num_additional_bitsの値が0又は<<1>>6に等しいことが要求されているが、本ドキュメントの当該バージョンに準拠するデコーダは、<<1>>6より大きいgci_num_additional_bitsの値が構文に存在することを許可し、且つgci_num_additional_bitsが<<1>>6より大きい場合、すべてのgci_reserved_zero_bit [i] 構文要素の値を無視すべきである。gci_num_additional_bitsが0に等しい場合、additional_general_constraint_s_info() 構文構造は、いかなる制約もかけない。

20

【0109】

別の例では、共通制約情報構文要素のセマンティックは、次のように変更される。

30

【0110】

gci_num_additional_bitsは、共通制約情報構文構造内の gci_alignment_zero_bit 構文要素（存在する場合）以外の追加 GCI ビットの数に規定する。本ドキュメントの当該バージョンに準拠するビットストリームでは、gci_num_additional_bitsの値は、0又は<<1>>6に等しくなるべきである。gci_num_additional_bitsの<<1>>6より大きい値は、ITU-T | ISO / IECに予約されて将来の使用に用いられる。本ドキュメントの当該バージョンでは、gci_num_additional_bitsの値が0又は<<1>>6に等しいことが要求されているが、本ドキュメントの当該バージョンに準拠するデコーダは、<<1>>6より大きいgci_num_additional_bitsの値が構文に存在することを許可し、且つgci_num_additional_bitsが<<1>>6より大きい場合、すべてのgci_reserved_zero_bit [i] 構文要素の値を無視すべきである。存在しない場合、additional_general_constraint_s_info() 構文構造は、いかなる制約もかけない。

40

【0111】

共通制約情報フラグを初期化する別の実施例では、gci_present_flag が1に等しく、且つgci_num_additional_bitsが0に等しい場合、gci_all_rap_pictures_constraint_flag、gci

50

`no_extended_precision_processing_constraint_flag`、`gci_no_ts_residual_coding_rice_constraint_flag`、`gci_no_reverse_last_sing_coeff_constraint_flag`、`gci_no_rrc_rice_extension_constraint_flag`、及び `gci_no_persistent_rice_adaptation_constraint_flag` は、これらのフラグ各自のセマンティックによって規定された対応する制約を常にかけることができる。言い換えると、`GCI` フラグが存在し、且つ `gci_num_additional_bits` が 0 に等しい場合、これら 6 つの `GCI` フラグが 1 に等しいと推測される。

【 0 1 1 2 】

一例として、`gci_num_additional_bits`のセマンティックの可能な変化は、次に示す通りであり、これらの変化は、VVCバージョン2に定められた現在の新規追加内容に基づくものである。

【 0 1 1 3 】

`gci_num_additional_bits`は、共通制約情報構文構造内の`gci_alignment_zero_bit`構文要素（存在する場合）以外の追加GCIビットの数を規定する。本ドキュメントの当該バージョンに準拠するビットストリームでは、`gci_num_additional_bits`の値は、0又は1に等しくなるべきである。`gci_num_additional_bits`の1より大きい値は、ITU-T | ISO / IECに予約されて将来の使用に用いられる。本ドキュメントの当該バージョンでは、`gci_num_additional_bits`の値が0又は1に等しいことが要求されているが、本ドキュメントの当該バージョンに準拠するデコーダは、1より大きい`gci_num_additional_bits`の値が構文に存在することを許可し、且つ`gci_num_additional_bits`が1より大きい場合は、すべての`gci_reserved_zero_bit[i]`構文要素の値を無視すべきである。`gci_present_flag`が1に等しく、且つ`gci_num_additional_bits`が0に等しい場合、`gci_all_rap_pictures_constraint_flag`、`gci_no_extended_precision_processing_constraint_flag`、`gci_no_t_s_residual_coding_rice_constraint_flag`、`gci_no_reverse_last_sig_coeff_constraint_flag`、`gci_no_rrc_rice_extension_constraint_flag`、及び`gci_no_persistent_rice_adaptation_constraint_flag`がすべて1に等しいと推測される。

【 0 1 1 4 】

別の例として、`gci_num_additional_bits`のセマンティックの可能な変化は、次に示す通りであり、これらの変化は、VVCバージョン2に定められた現在の新規追加内容に基づくものである。

【 0 1 1 5 】

`gci_num_additional_bits`は、共通制約情報構文構造内の`gci_alignment_zero_bit`構文要素（存在する場合）以外の追加GCIビットの数を規定する。本ドキュメントの当該バージョンに準拠するビットストリームでは、`gci_num_additional_bits`の値は、0又は $\ll 1 \gg 6$ に等しくなる。`gci_num_additional_bits`の $\ll 1$ より大きい $\gg 0$ 及び6以外の値は、ITU-T | ISO / IECに予約されて将来の使用に用いられる。本ドキュメントの当該バージョンでは、`gci_num_additional_bits`の値が0又は $\ll 1 \gg 6$ に等しいことが要求されているが、本ドキュメントの当該バージョンに準拠するデコーダは、 $\ll 1$ より大きい $\gg 0$ 及び6以外の`gci_num_additional_bits`の値が構文に存在することを許可し、且つ`gci_num_additional`

onal_bitsが《1より大きい》0及び6以外である場合、すべてのgci_reserved_zero_bit[i]構文要素の値を無視すべきである。gci_present_flagが1に等しく、gci_num_additional_bitsが0に等しい場合、gci_all_rap_pictures_constraint_flag、gci_no_extended_precision_processing_constraint_flag、gci_no_ts_residual_coding_rice_constraint_flag、gci_no_reverse_last_sig_coeff_constraint_flag、gci_no_rrc_rice_extension_constraint_flag、及びgci_no_persistent_rice_adaptation_constraint_flagがすべて1に等しいと推測する。

10

【0116】

上述の実施例に加えて、又は上述の実施例の補足又は代替として、誤解を招く構文要素gci_reserved_zero_bit[i]をgci_reserved_bit[i]に名称変更することができる。例えば、変更後の構文とセマンティックは、次の通りであり得る。

【0117】

20

30

40

50

【表 8】

general_constraints_info() {	記述子
gci_present_flag	u(1)
if(gci_present_flag) {	
/* 共通general */	
gci_intra_only_constraint_flag	u(1)
...	
gci_num_additional_bits	u(8)
if(gci_num_additional_bits > 05) {	
gci_all_rap_pictures_constraint_flag	u(1)
gci_no_extended_precision_processing_constraint_flag	u(1)
gci_no_ts_residual_coding_rice_constraint_flag	u(1)
gci_no_rrc_rice_extension_constraint_flag	u(1)
gci_no_persistent_rice_adaptation_constraint_flag	u(1)
gci_no_reverse_last_sig_coeff_constraint_flag	u(1)
numAdditionalBitsUsed = 6	
} else	
numAdditionalBitsUsed = 0	
for(i = 0; i < gci_num_additional_bits - numAdditionalBitsUsed; i++)	
~~gci_reserved_zero_bit[i]~~ gci_reserved_bit[i]	u(1)
}	
while(!byte_aligned())	
gci_alignment_zero_bit	f(1)
}	

10

20

30

【0118】

《gci_reserved_zero_bit[i]》gci_reserved_bit[i]は、任意の値でもよい。その存在と値は、当該仕様のこのバージョンで規定された復号プロセスには影響しない。本仕様の当該バージョンに準拠するデコーダは、すべての《gci_reserved_zero_bit[i]》gci_reserved_bit[i]構文要素の値を無視すべきである。

40

【0119】

さらなる実施例では、構文要素 gci_reserved_zero_bit[i] の名称を gci_reserved_bit[i] に変更する。例えば、変更後の構文とセマンティックは、次の通りであり得る。

【0120】

50

【表 9】

general_constraints_info() {	記述子
gci_present_flag	u(1)
if(gci_present_flag) {	
/* 共通general */	
gci_intra_only_constraint_flag	u(1)
...	
gci_num_additional_bits	u(8)
if(gci_num_additional_bits > 0 5) {	
gci_all_rap_pictures_constraint_flag	u(1)
gci_no_extended_precision_processing_constraint_flag	u(1)
gci_no_ts_residual_coding_rice_constraint_flag	u(1)
gci_no_rrc_rice_extension_constraint_flag	u(1)
gci_no_persistent_rice_adaptation_constraint_flag	u(1)
gci_no_reverse_last_sig_coeff_constraint_flag	u(1)
numAdditionalBitsUsed = 6	
} else	
numAdditionalBitsUsed = 0	
for(i = 0; i < gci_num_additional_bits - numAdditionalBitsUsed; i++)	
~~gci_reserved_zero_bit[i]~~ gci_reserved_bit[i]	u(1)
}	
while(!byte_aligned())	
gci_alignment_zero_bit	f(1)
}	

10

20

30

【0121】

《gci_reserved_zero_bit[i]》gci_reserved_bit[i]は、任意の値でもよい。その存在と値は、当該仕様のこのバージョンで規定された復号プロセスには影響しない。本仕様の当該バージョンに準拠するデコーダは、すべての《gci_reserved_zero_bit[i]》gci_reserved_bit[i]構文要素の値を無視すべきである。

40

【0122】

共通制約フラグについて

gci_all_rap_pictures_constraint_flagについて

gci_all_rap_pictures_constraint_flagは、ピクチャをIRAP又はGDRピクチャに制限することを示すために使用される。

【0123】

50

NAL (Network Abstraction Layer) は、VVC 構文要素を「NAL ユニット」として編成するシステムインターフェースである。このような構造により、VVC を簡単かつ効果的にカスタマイズでき、様々なユースケース（リアルタイム通信アプリケーションからストレージアプリケーションのファイルフォーマットまで）に適合する。VVC 標準における NAL ユニットタイプの完全なリストは、次の表の通りである。

【 0 1 2 4 】

【表 1 0 (1) 】

nal__unit__type	nal__unit__type の名称	NAL ユニットと RBSP 構文構造の内容	NAL ユニットタイプクラス
0	TRAIL__NUT	後続ピクチャ又はサブピクチャの符号化スライス * slice__layer__rbsp()	VCL
1	STSA__NUT	STSA ピクチャ又はサブピクチャの符号化スライス * slice__layer__rbsp()	VCL
2	RADL__NUT	RADL ピクチャ又はサブピクチャの符号化スライス * slice__layer__rbsp()	VCL
3	RASL__NUT	RASL ピクチャ又はサブピクチャの符号化スライス * slice__layer__rbsp()	VCL
4 . . 6	RSV__VCL__ 4 . . RSV__VCL__ __6	予約された非 IRAP VCL NAL ユニットタイプ	VCL
7 8	IDR__W__RADL IDR__N__LP	IDR ピクチャ又はサブピクチャの符号化スライス * slice__layer__rbsp()	VCL
9	CRA__NUT	CRA ピクチャ又はサブピクチャの符号化スライス * slice__layer__rbsp()	VCL
10	GDR__NUT	GDR ピクチャ又はサブピクチャの符号化スライス * slice__layer__rbsp()	VCL

10

20

30

40

50

【表 10 (2)】

11	RSV__IRAP__1 1	予約されたIRAP VCL NALユニットタイプ	VCL
12	OPI__NUT	オペレーティングポイント情報 operating__point__information__rb sp()	非VCL
13	DCI__NUT	復号能力情報 decoding__capability__information __rbsp()	非VCL
14	VPS__NUT	ビデオパラメータセット video__parameter__set__rbsp()	非VCL
15	SPS__NUT	シーケンスパラメータセット seq__parameter__set__rbsp()	非VCL
16	PPS__NUT	ピクチャパラメータセット pic__parameter__set__rbsp()	非VCL
17 18	PREFIX__APS __NUT SUFFIX__APS __NUT	適応パラメータセット adapation__paramater__set__rbsp ()	非VCL
19	PH__NUT	ピクチャヘッダー picture__head__rbsp()	非VCL
20	AUD__NUT	AUデリミタ access__unit__delimiter__rbsp()	非VCL
21	EOS__NUT	シーケンスの終了 end__of__seq__rbsp()	非VCL
22	EOB__NUT	ビットストリームの終了 end__of__bitstream__rbsp()	非VCL

10

20

30

40

50

【表 10 (3)】

23 24	PREFIX__SEI__ NUT SUFFIX__SEI__ NUT	補足拡張情報(Supplemental enhancement information) sei__rbsp()	非VCL
25	FD__NUT	充填データ filler__data__rbsp()	非VCL
26 27	RSV__NVCL__2 6 RSV__NVCL__2 7	予約済みの非VCL NALユニットタイプ	非VCL
28 . . 3 1	UNSPEC__2 8 . . UNSPEC__ 31	未指定の非VCL NALユニットタイプ	非VCL
* pps__mixed__nalu__types__in__pic__flagが0に等しい場合のピクチャの属性を示し、及びpps__mixed__nalu__types__in__pic__flagが1に等しい場合のサブピクチャの属性を示す。			

10

20

【0125】

30

ビデオ符号化レイヤ (VCL) として分類されるNALユニットは、低レベル構文要素を含み、非VCLとして分類されるNALユニットは、高レベル構文要素を含む。ビデオシーケンスのピクチャは、VCL NALユニットから復号される。異なるタイプのVCLカテゴリは、高レベルで依存関係を示す。例えば、TRAIL__NUT (0) からRSV__VCL__6 (6) までのNALユニットは通常、インター予測ツールを使用することができ、これは、前に復号された (参照) ピクチャへのアクセスができるかどうかによって依存する。イントラ予測ツールのみを使用して圧縮されたピクチャと比較して、インター予測ツールを使用して符号化されたピクチャは、より効率的に圧縮されることができる。しかし、参照ピクチャを取得することができない場合では、このような復号依存性に問題が生じる。

40

【0126】

イントラランダムアクセスポイント (IRAP) ピクチャは、符号化されたピクチャであり、そのすべてのVCL NALユニットのnal__unit__typeの値は同じであり、IDR__W__RADLからCRA__NUTまでの範囲 (IDR__W__RADL及びCRA__NUTを含む) 内である。IRAPピクチャは、CRAピクチャであってもよいし、IDRピクチャであってもよい。IRAPピクチャは、復号プロセスで同じレイヤの参照ピクチャからのインター予測を使用しない。ビットストリーム内の復号順における1番目のピクチャは、IRAPピクチャ又は漸次復号リフレッシュ (GDR) ピクチャである。シングルレイヤービットストリームの場合、参照する必要があるときに、必要なパラメータセットを取得することができれば、IRAPピクチャと、CLVS内の復号順で

50

配列されたすべての後続の非 R A S L ピクチャを正しく復号することができ、復号順が I R A P ピクチャの前にあるいずれかのピクチャに対して復号プロセスを実行する必要がない。I R A P ピクチャの `pps__mixed__nal__types__in__pic__flag` の値は 0 に等しい。あるピクチャの `pps__mixed__nal__types__in__pic__flag` が 0 に等しく、且つ当該ピクチャの任意のスライス (`slice`) の `nal__unit__type` が、I D R __ W __ R A D L から C R A __ N U T までの範囲 (I D R __ W __ R A D L 及び C R A __ N U T を含む) 内にある場合、当該ピクチャの他のすべてのスライスは同じ `nal__unit__type` 値を有し、1 番目のスライスを受信した後、当該ピクチャが I R A P ピクチャであると既知している。

【0127】

10

したがって、I R A P ピクチャは、同じレイヤにまたがるインター予測を使用しない。当該制限は、I R A P ピクチャが、ストリーミングビデオアプリケーションのエラー回復ポイント、又はビデオオンデマンド再生アプリケーションのビデオの検索ポイントとして使用できる。しかし、I R A P ピクチャの圧縮効率は、通常、非 I R A P ピクチャの圧縮効率より低い。

【0128】

漸次復号リフレッシュ (G D R) ピクチャは、非 I R A P ピクチャと I R A P ピクチャとの間のトレードオフ (`trade-off`) として V V C 標準に導入された。G D R ピクチャは、インター予測を使用しない「クリーン (`clean`) 」な部分があり、ピクチャの残りの部分は、インター予測を自由に使用できる。このようにピクチャを分割することで、パケット損失などのエラーイベントが発生した場合でも、「クリーン」な部分は正しく復号される。連続する G D R ピクチャでは、「クリーン」な部分の空間位置が回転されるため、最終的にはピクチャ全体がエラーから回復することができる。

20

【0129】

再生の柔軟性のストリーミング回復力が重要なビデオアプリケーションでは、すべてのピクチャを I R A P ピクチャ又は G D R ピクチャに制限することが望ましい場合がある。V V C v 2 では、G C I フラグ `gci__all__rap__pictures__constraint__flag` が導入され、高レベルでこのような制限を示すことができる。 `gci__all__rap__pictures__constraint__flag` が 1 に等しい場合、O l s I n S c o p e 内のすべてのピクチャが `ph__recovery__poc__cnt` が 0 に等しい G D R ピクチャ、又は I R A P ピクチャであることを規定する。 `gci__all__rap__pictures__constraint__flag` が 0 に等しい場合、このような制約をかけない。 `gci__all__rap__pictures__constraint__flag` が存在しない場合、その値が 0 に等しいと推測される。

30

【0130】

`profile__tier__level` () 構文構造が V P S に含まれている場合、O l s I n S c o p e は、V P S によって規定された 1 つ又は複数の出力レイヤセット (O L S) である。 `profile__tier__level` () 構文構造が S P S に含まれている場合、O l s I n S c o p e は、S P S を参照するレイヤのうち最下位レイヤのみを含む O L S であり、当該最下位レイヤは独立したレイヤである。

40

【0131】

`gci__no__extended__precision__processing__constraint__flag`

G C I フラグ `gci__no__extended__precision__processing__constraint__flag` は、変換精度拡張の V V C v 2 ツールが制約されているか否かを高レベルで伝送する。V V C 標準では、ビデオ信号の画素は、整数値で表される。V V C 標準に記載されたすべての計算と処理は、整数演算で表現される。複雑さと相互運用性を考慮し、この制限は重要である。まず、同等の浮動小数点演算と比較して、整数演算 (加算、乗算、除算) の計算コストは通常低くなる。次に、浮動小数点演算は確定的に標準化されていない。浮動小数点の加算と乗算は、必ずしも交換法則を満たす

50

ことができない（例えば、 $(a + b) + c$ は必ずしも $a + (b + c)$ と等しいとは限らない）、異なるプラットフォームでの浮動小数点演算の結果が同じであるとは保証されない。

【0132】

ビデオ信号サンプルのビット深度は、ビデオソースの属性であり、VVC標準では、`BitDepth`と表記されている。ハイブリッドビデオ符号化システムでは、ビデオサンプルは、インター予測ツール又はイントラ予測ツールによって予測される。オリジナルのビデオサンプルと予測サンプルの差は残差と呼ばれる。最悪のケースでは、これらの残差係数のビット深度は、拡大される（例えば、`BitDepth + 1`）可能性があるが、VVC標準では、これらの残差係数は、`BitDepth`のビット深度を維持するためにトリミングされる。実際には、最悪のケースが発生することがありなく、実際のエンコーダは、オリジナルのビデオ信号より大きい残差を生成する予測ツールを選択しないためである。

10

【0133】

その後、通常、整数化された離散コサイン変換（DCT）によって残差係数を分析し、変換係数を生成する。離散コサイン変換（DCT）は、線形の可逆関数であり、形式的には次のように表される。

【0134】

【数1】

$$f: \mathbb{R}^N \rightarrow \mathbb{R}^N$$

20

ここで、

【数2】

$$\mathbb{R}^N$$

は、実数のN次元空間を表す。VVC標準では、DCTの整数化された近似（即ち、

【数3】

$$f: \mathbb{Z}^N \rightarrow \mathbb{Z}^N$$

30

）が使用される。予測とは異なり、変換の精度が符号化ゲインに影響するため、通常、変換段階でビット深度のある程度の拡張は許可される。近似されたDCT係数のビット深度と、結果として得られた変換係数のビット深度は、ハードウェアの複雑さと符号化性能をトレードオフする設計である。

【0135】

VVCv1では、変換係数のビット深度は16である。つまり、各変換係数は、 $[-2^{15}, 2^{15} - 1]$ の範囲内の値を取る。ビデオサンプルを整数化されたDCT係数で乗算すると、通常、ビット深度が16より大きい中間変換係数が生成される。所望のビット深度を有する変換係数を生成するために、中間変換係数を右にビットシフトする。このような動作は、本質的に損失を伴う。

40

【0136】

VVCv2では、SPSフラグ`sps__extended__precision__flag`を1に設定することによって、拡張変換精度が有効になる。拡張変換精度が有効になると、変換係数のビット深度は、 $(\text{Log2TransformRange} + 1)$ に増加する。正確な変換係数のビット深度は、`BitDepth`に依存する。`sps__extended__precision__flag`が1に等しい場合、スケーリングと変換プロセスにおいて拡張ダイナミックレンジを変換係数に使用すること、及び拡張ダイナミックレンジを`abs__remaining []`と`dec__abs__level []`構文要素の二

50

値化に使用することを規定する。sps_extended_precision_flag が 0 に等しい場合、スケーリングと変換プロセスにおいて拡張ダイナミックレンジが使用されないこと、及び abs_remaining[] と dec_abs_level[] 構文要素の二値化に拡張ダイナミックレンジを使用しないことを規定する。存在しない場合、sps_extended_precision_flag の値が 0 に等しいと推測される。変数 Log2TransformRange は、次のように導出される。

【0137】

【数 4】

$$\text{Log2TransformRange} = \text{sps_extended_precision_flag} ? \text{Max}(15, \text{Min}(20, \text{BitDepth} + 6)) : 15$$

10

$$\text{CoeffMin} = -(1 \ll (\text{sps_extended_precision_flag} ? \text{Max}(15, \text{Min}(20, \text{BitDepth} + 6)) : 15))$$

$$\text{CoeffMax} = (1 \ll (\text{sps_extended_precision_flag} ? \text{Max}(15, \text{Min}(20, \text{BitDepth} - 6)) : 15)) - 1$$

gci_no_extended_precision_processing_constraint_flag が 1 に等しい場合、OlsInScope 内のすべてのピクチャの sps_extended_precision_flag が 0 に等しいことを規定する。本明細書では、OlsInScope は、「スコープ内出力レイヤセット (in-scope output layer set)」とも呼ばれる。profile_tier_level() 構文構造が VPS に含まれている場合、OlsInScope は、VPS によって規定された 1 つ又は複数の出力レイヤセット (OLS) である。profile_tier_level() 構文構造が SPS に含まれている場合、OlsInScope は、SPS を参照するレイヤのうち最下位レイヤのみを含む OLS であり、当該最下位レイヤは独立したレイヤである。gci_no_extended_precision_processing_constraint_flag が 0 に等しい場合、このような制約をかけない。gci_no_extended_precision_processing_constraint_flag が存在しない場合、その値が 0 に等しいと推測される。

20

30

【0138】

gci_no_ts_residual_coding_rice_constraint_flag について

GCI フラグ gci_no_ts_residual_coding_rice_constraint_flag は、高レベルで伝送され、明示的なライスパラメータで VVCv2 ツールが制約されているか否かを示す。エントロピ符号化では、各構文要素の値は、エントロピ符号化プロセスによってビットシーケンスに符号化され、ビットストリームに挿入される。

【0139】

VVC で使用される 2 つのエントロピ符号化プロセスは、コンテキスト適応型バイナリ算術コーディング (CABAC) とライスコーディングである。CABAC エンジンには、自己適応性を有し、構文要素を理論的なシャノン限界に非常に近いビットレートに圧縮することができる。ただし、算術コーディングが複雑である。CABAC を使用して非バイナリ構文要素 (例えば、値の範囲が 0 又は 1 を超える構文要素) を符号化するために、まず、構文要素を「bin」セットに二値化する。以下の表に、二値化の 2 つの例を示す。2 番目の例は、可変長の二値化の場合、構文要素の一部の値は対応する bin がないことを示す。

40

【0140】

50

【表 1 1】

表:固定長の二値化の例

値	二値化
0	00
1	01
2	10
3	11

10

【0 1 4 1】
【表 1 2】

表:可変長の二値化の例

値	二値化
0	0
1	10
2	110
3	1110
4	1111

20

【0 1 4 2】

二値化の各 bin は、CABAC エンジンによって符号化されることができる。ただし、CABAC を使用して bin を符号化するために、エンジンは、関連する「コンテキスト」を記憶し、更新する必要がある。該コンテキストは、bin の確率分布をモデル化する。構文要素の二値化では、各 bin には単独のコンテキストが必要である。

30

【0 1 4 3】

値の範囲が広い構文要素は単に CABAC を利用して符号化することを望まなく、なぜなら、このような構文要素は二値化が長いためである。コンテキストの記憶と更新のために過度のオーバーヘッドが引き起こすためである。例えば、残差係数値は、単に CABAC を利用して符号化することが不便である。CABAC とは対照的に、ライスコーディングは、コンパクトなパラメータを使用して、非バイナリ値の確率分布をモデル化することができる。

【0 1 4 4】

ライスコーディングは、ゴロムコーディング、又はライスーゴロムコーディングとも呼ばれる。ゴロムコーディングは、単一のパラメータ M によって制御されるエントロピコードであり、パラメータ M は、正の整数に制限される。ライスコーディングは、ゴロムコーディングのサブセットであり、ここで、エントロピコードは、単一のライスパラメータ R によって制御され、R が非負整数である。ライスパラメータ R を有するライスコード (Rice code) は、ゴロムコードと同等であり、ここで、 $M = 2^R$ である。

40

【0 1 4 5】

非負整数値 x を、次のようにライスパラメータ R を有するライスコードに二値化する。商 q と剰余 r は、次のように計算される。

【0 1 4 6】

50

【数 5】

$$q = \text{floor} \left(\frac{x}{2^R} \right)$$

$$r = x \bmod 2^R$$

x のライスコードは、プレフィックスコード (Prefix Code) とサフィックスコード (Suffix Code) の組み合わせである。プレフィックスコードは、q の一進法コードによって決定される。例えば、VVC で使用されるプレフィックスコードは、切り詰められた一進法コード (Truncated Unary Code) である。

【0 1 4 7】

【表 1 3】

q	一進法コード
0	0
1	10
2	110
3	1110
4	11110
5	111110
6	111111

【0 1 4 8】

一進法コードが切り詰められたものであるため、ライスコードも切り詰められ、これは、符号化可能な x の値の範囲は限られていることを意味する。VVC では、切り詰められたライスコードは、

【数 6】

$$\left[0, 6 * 2^R \right]$$

の範囲の x の値に適用される。

【0 1 4 9】

サフィックスコードの長さは、R ビットの r の固定長コードを二値化したものであり、例えば、R = 3 である場合、サフィックスコードは、次の通りである

【0 1 5 0】

【表 1 4】

r	固定長コード
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

10

【0 1 5 1】

VVCでは、残差係数は、CABAC、ライスコーディング及び指数ゴロムコーディングの組み合わせによって符号化される。数少ない構文要素フラグが定義されており、これらのフラグは小さいマグニチュードの残差の値を伝送するのに十分である。例えば、`sig__coeff__flag`は、残差のマグニチュードがゼロか否かを示す。`sig__coeff__flag`が1である場合、さらにフラグを伝送することができ（フラグは、通常、`abs__level__gt__flag`と命名される）、これらのフラグは、残差のマグニチュードが1、2、3などより大きいかなを示す。これらのフラグは、CABACエンジンによってコンテキスト符号化される。残差係数のほとんどはマグニチュードが小さいため、CABACによって、比較的少数のbinを使用して効率的に大部の残差を符号化できる。

20

【0 1 5 2】

残差係数フラグによって伝送できない残差係数の任意の残りのマグニチュードは、構文要素`abs__remainder`で伝送される。`abs__remainder`の値が 6×2^R 以下である場合、当該構文要素は、`abs__remainder`のライスコードによって完全に伝送される。 6×2^R より大きい場合、当該構文要素は、 6×2^R のライスコードと

30

【数 7】

$$\left(\text{abs_remainder} - 6 \times 2^R \right)$$

40

の指数ゴロムコードの組み合わせによって伝送される。ここで、指数ゴロムコーディングプロセスについては説明しない。

【0 1 5 3】

ライスコーディングはCABACより簡単であるが、適切な条件では、ライスコーディングも低ビットレートの圧縮に効果的である。残差係数がすべて小さい値である場合、小さいライスパラメータを有するライスコーディングの方がより効率的である。逆に、残差係数の一部の値が大きい場合、より大きなライスパラメータがより適している場合がある。残差係数の統計データに適應させるために、ライスパラメータは、`locSumAbs`値に基づいて自己適應的に決定され、`locSumAbs`値は、隣接残差係数のマグニチュードに基づいて計算される。

50

【 0 1 5 4 】

【 表 1 5 】

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

【 0 1 5 5 】

自己適応ライスパラメータの決定は、「レギュラ残差コーディング」(RRC)における残差係数のために設計され、これは、離散コサイン変換(DCT)を実行することによって得られた残差係数である。しかし、VVcv1では、それは、「変換スキップ残差コーディング」(TSRC)における残差係数にも適用される。VVcv2では、ライスパラメータを決定するための代替メカニズムが、変換スキップ係数に有益である可能性があることに気づいている。

【 0 1 5 6 】

VVcv2における代替メカニズムは、スライスレベルの構文要素sh_ts_residual_coding_rise_idx_minus1によってライスパラメータを明示的に伝送することを許可する。当該構文要素を伝送するときに、ライスパラメータは、 $R = sh_ts_residual_coding_rise_idx_minus1 + 1$ に設定される。ライスパラメータの当該値は、スライスの持続時間内に保持される。

20

【 0 1 5 7 】

【 表 1 6 】

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(!sh_ts_residual_coding_disabled_flag && sps_ts_residual_coding_rice_present_in_sh_flag)	
sh_ts_residual_coding_rice_idx_minus1	u(3)
if(sps_reverse_last_sig_coeff_enabled_flag)	
sh_reverse_last_sig_coeff_flag	u(1)
if(pps_slice_header_extension_present_flag) {	
sh_slice_header_extension_length	ue(v)

30

40

【 0 1 5 8 】

sh_ts_residual_coding_rice_idx_minus1+1は、現在のスライス内のresidual_ts_coding()構文構造に使用されるライスパラメータを規定する。存在しない場合、sh_ts_residual_coding_rice_idx_minus1の値が0に等しいと推測される。

【 0 1 5 9 】

代替メカニズムを有効にするか否かは、SPSレベルのフラグsps_ts_residual_coding_rice_present_in_sh_flagによって制

50

御される。当該フラグが0に設定されている場合、代替ライスパラメータ伝送を有効にしない。GCIフラグ `gci_no_ts_residual_coding_rice_constraint_flag` は、明示的なライスパラメータ伝送のVVCv2ツールが制約されているか否かを高レベルで伝送する。`gci_no_ts_residual_coding_rice_constraint_flag` が1に等しい場合、OlsInScope内のすべてのピクチャの `sps_ts_residual_coding_rice_present_in_sh_flag` が0に等しいことを規定する。`gci_no_ts_residual_coding_rice_constraint_flag` が0に等しい場合、このような制約をかけない。`gci_no_ts_residual_coding_rice_constraint_flag` が存在しない場合、その値が0に等しいと推測される。

10

【0160】

`gci_no_rrc_rice_extension_constraint_flag` は、OlsInScope内のすべてのピクチャの `sps_rrc_rice_extension_flag` を規定する。`gci_no_rrc_rice_extension_constraint_flag` が1に等しい場合、OlsInScope内のすべてのピクチャの `sps_rrc_rice_extension_flag` が0に等しいことを規定する。`gci_no_rrc_rice_extension_constraint_flag` が0に等しい場合、このような制約をかけない。

20

【0161】

高ビット深度と高ビットレートアプリケーションでは、RRCの多くの位置に多くの大きな量子化レベルが存在する。このようなアプリケーションでは、大きいライスパラメータは、残りのレベルを表すために必要なbin数が少なくなる。VVCv1におけるライスパラメータの導出方式は、VVCv2アプリケーションには最適ではない可能性がある。よって、代替ライスパラメータ導出を採用することができ、これは、`sps_rrc_rice_extension_flag` によって伝送できる。`sps_rrc_rice_extension_flag` が1に等しい場合、`abs_remaining[]` と `dec_abs_level[]` の二値化に代替ライスパラメータ導出を使用することを規定する。`sps_rrc_rice_extension_flag` が0に等しい場合、`abs_remaining[]` と `dec_abs_level[]` の二値化に代替ライスパラメータ導出を使用しないことを規定する。存在しない場合、`sps_rrc_rice_extension_flag` の値が0に等しいと推測される。

30

【0162】

以下は、VVCv2における `sps_rrc_rice_extension_flag` を使用してライスパラメータを決定する例を示す。コンポーネントインデックス `cIdx` を有する変換ブロックの配列 `AbsLevel[x][y]` と左上隅の輝度位置 `(x0, y0)` が与えられると、以下の擬似コードプロセスのルールに従って変数 `locSumAbs` を導出する（下線が付けられている部分は、VVCv1に対するVVCv2の追加された内容である）。

40

【0163】

【数8】

```

locSumAbs = 0
if( xC < ( 1 << AbsLlog2TbWidth ) - 1 ) {
  locSumAbs -= AbsLevel[ xC + 1 ][ yC ]
  if( xC < ( 1 << AbsLlog2TbWidth ) - 2 )
    locSumAbs -= AbsLevel[ xC + 2 ][ yC ]
  else
    locSumAbs -= HistValue
  if( yC < ( 1 << AbsLlog2TbHeight ) - 1 )
    locSumAbs -= AbsLevel[ xC + 1 ][ yC + 1 ]
  else
    locSumAbs -= HistValue
} else
  locSumAbs -= 2 * HistValue
if( yC < ( 1 << AbsLlog2TbHeight ) - 1 ) {
  locSumAbs -= AbsLevel[ xC ][ yC + 1 ]
  if( yC < ( 1 << AbsLlog2TbHeight ) - 2 )
    locSumAbs -= AbsLevel[ xC ][ yC + 2 ]
  else
    locSumAbs -= HistValue
} else
  locSumAbs -= HistValue

```

10

20

リスト T_x 「」と R_x 「」は、次のように規定される。

【 0 1 6 4 】

【 数 9 】

$T_x[] = \{ 32, 128, 512, 2048 \}$ (1523)

$R_x[] = \{ 0, 2, 4, 6, 8 \}$ (1524)

変数 $shiftVal$ の値は、次のように導出される。

【 0 1 6 5 】

【 数 1 0 】

30

```

if( !sps_ rrc_ rice_ extension_ flag )
  shiftVal = 0
else
  shiftVal = ( localSumAbs < Tx[ 0 ] ) ? Rx[ 0 ] : ( ( localSumAbs < Tx[ 1 ] ) ? Rx[ 1 ] :
    ( ( localSumAbs < Tx[ 2 ] ) ? Rx[ 2 ] : ( ( localSumAbs < Tx[ 3 ] ) ? Rx[ 3 ] :
      Rx[ 4 ] ) ) )

```

$locSumAbs$ の値は、次のように更新される。

【 0 1 6 6 】

【 数 1 1 】

40

$locSumAbs = Clip3(0, 31, (locSumAbs \gg shiftVal) - baseLevel * 5)$ (1526X2)

変数 $locSumAbs$ が与えられると、まず表 128 に従ってライスパラメータ $cRiceParam$ を導出し、次に、 $cRiceParam$ は次のように更新される。

【 0 1 6 7 】

【 数 1 2 】

$cRiceParam = cRiceParam + shiftVal$ (1526X3)

50

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

【0168】

【数13】

$$\text{ZeroPos}[n] = (\text{QState} < 2 ? 1 : 2) \ll \text{cRiceParam} \quad (1518)$$

【0169】

【表17】

表128—`locSumAbs`に基づく`cRiceParam`の仕様

10

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

【0170】

式 (1526X3) から、`VVCv1` と比較して、`VVCv2` で絶対レベルの残りの部分を二値化するために使用される `cRiceParam` はもっと大きくしても良いことが分かる。

20

【0171】

`gci_no_persistent_rice_adaptation_constraint_flag` について

`gci_no_persistent_rice_adaptation_constraint_flag` は、以前の TU 状態を使用した二値化のためのライスパラメータの導出に対する制約を伝送する。`gci_no_persistent_rice_adaptation_constraint_flag` が 1 に等しい場合、`OlsInScope` 内のすべてのピクチャの `sps_persistent_rice_adaptation_enabled_flag` が 0 に等しいことを規定する。`gci_no_persistent_rice_adaptation_constraint_flag` が 0 に等しい場合、このような制約をかけない。`gci_no_persistent_rice_adaptation_constraint_flag` が存在しない場合、その値が 0 に等しいと推測される。

30

【0172】

`sps_persistent_rice_adaptation_enabled_flag` が 1 に等しい場合、各 TU の開始時に、以前の TU から蓄積された統計データを使用して、`abs_remainder[]` と `dec_abs_level[]` の二値化のためのライスパラメータ導出を初期化することを規定する。`sps_persistent_rice_adaptation_enabled_flag` が 0 に等しい場合、ライスパラメータ導出において以前の TU 状態が使用されないことを規定する。存在しない場合、`sps_persistent_rice_adaptation_enabled_flag` の値が 0 に等しいと推測される。以下は、`VVCv2` で `sps_persistent_rice_adaptation_enabled_flag` を使用して、ライスパラメータを決定する例を示す（下線が付けられている部分は、`VVCv1` に対する `VVCv2` の追加された内容である）。

40

【0173】

CTU がスライス又はタイル (`tile`) 内の一番目の CTU である場合、サブ条項 9.3.2.2 の規定に従ってコンテキスト変数の初期化プロセスを呼び出し、配列 `Pre`

50

`dictorPaletteSize[chType]` (`chType=0, 1`) を 0 に初期化し、配列 `StatCoeff[i]` ($i=0..2$) を次のように初期化する。

【0174】

【数14】

$\text{StatCoeff}[i] = \text{sps_persistent_rice_adaptation_enabled_flag} \cdot 2 \cdot$

$\text{Floor}(\text{Log2}(\text{BitDepth} - 10)) : 0$

(1513X3)

`StatCoeff[i]` は、`HisValue` を計算するために使用され、`HisValue` は、式 (1517) の `locSumAbs` を計算するために使用され、`HisValue` は、TUごとに1回更新されることができる。`HisValue` によってブロック境界に位置するライスパラメータの導出値がより正確になる。

10

【0175】

`gci__no__reverse__last__sig__coeff__constraint__flag` について

`gci__no__reverse__last__sig__coeff__constraint__flag` が 1 に等しい場合、`OlsInScope` 内のすべてのピクチャの `sps__reverse__last__sig__coeff__enabled__flag` が 0 に等しいことを規定する。`gci__no__reverse__last__sig__coeff__constraint__flag` が 0 に等しい場合、このような制約をかけない。`gci__no__reverse__last__sig__coeff__constraint__flag` が存在しない場合、その値が 0 に等しいと推測される。

20

【0176】

レギュラ残差コーディング (RRC) では、TU内の最後の非ゼロレベルの位置 (x, y) は、最大4つの構文要素 `last__sig__coeff__x__prefix`、`last__sig__coeff__y__prefix`、`last__sig__coeff__x__suffix`、`last__sig__coeff__y__suffix` を使用して符号化される。`VVCv1` において現在 TU の (x, y) と (0, 0) との差を使用して当該位置を符号化する。これは、`VVCv1` にとって妥当であり、なぜなら、1つのTU内に多くのゼロレベルが存在し、ほとんどの非ゼロレベルはTUの左上隅に位置するためである。ただし、`VVCv2` アプリケーションにとっては、そうではない可能性があり、多くの非ゼロレベルがTU全体に広がるため、左上隅 (0, 0) ではなく右下隅を基準としてこの位置 (x, y) を符号化の方がより有益である可能性がある。`sh__reverse__last__sig__coeff__flag` は、このようなアプリケーションを処理するためのツールを提供する。

30

【0177】

`sh__reverse__last__sig__coeff__flag` が 1 に等しい場合、現在スライスの各変換ブロックについて、 $((\text{Log2ZotbWidth} < 1) - 1)$ 、 $((\text{Log2ZotbHeight} < 1) - 1)$ に対して、最後の重要係数 (`lastSignificantCoefficient`) の座標を符号化することを規定する。`sh__reverse__last__sig__coeff__flag` が 0 に等しい場合、現在スライスの各変換ブロックについて、(0, 0) に対して、最後の重要係数の座標を符号化することを規定する。存在しない場合、`sh__reverse__last__sig__coeff__flag` の値が 0 に等しいと推測される。

40

【0178】

スライスヘッダーにおいて、`sh__reverse__last__sig__coeff__flag` を利用して、`sps__reverse__last__sig__coeff__enabled__flag` に対して条件付きで次のように解析される。

【0179】

— `last__sig__coeff__x__suffix` が存在しない場合、

50

【数 1 5】

$$\text{LastSignificantCoeffX} = \text{last_sig_coeff_x_prefix} \quad (193)$$

－ そうでない場合 (last_sig_coeff_x_suffix が存在する場合)、

【数 1 6】

$$\begin{aligned} \text{LastSignificantCoeffX} = & (1 \ll ((\text{last_sig_coeff_x_prefix} \gg 1) - 1)) * \\ & (2 - (\text{last_sig_coeff_x_prefix} \& 1)) + \text{last_sig_coeff_x_suffix} \end{aligned} \quad (194)$$

10

sh_reverse_last_sig_coeff_flag が 1 に等しい場合、LastSignificantCoeffX の値は、次のように変更される。

【0 1 8 0】

【数 1 7】

$$\text{LastSignificantCoeffX} = (1 \ll \text{Log2ZoTbWidth}) - 1 - \text{LastSignificantCoeffX} \quad (195)$$

－ last_sig_coeff_y_suffix が存在しない場合、次の式が適用される。

20

【0 1 8 1】

【数 1 8】

$$\text{LastSignificantCoeffY} = \text{last_sig_coeff_y_prefix} \quad (196)$$

－ そうでない場合 (last_sig_coeff_y_suffix が存在する場合)、次の式が適用される。

【0 1 8 2】

【数 1 9】

$$\begin{aligned} \text{LastSignificantCoeffY} = & (1 \ll ((\text{last_sig_coeff_y_prefix} \gg 1) - 1)) * \\ & (2 - (\text{last_sig_coeff_y_prefix} \& 1)) + \text{last_sig_coeff_y_suffix} \end{aligned} \quad (197)$$

30

sh_reverse_last_sig_coeff_flag が 1 に等しい場合、LastSignificantCoeffY の値は、次のように変更される。

【0 1 8 3】

【数 2 0】

$$\text{LastSignificantCoeffY} = (1 \ll \text{Log2ZoTbHeight}) - 1 - \text{LastSignificantCoeffY} \quad (198)$$

40

【0 1 8 4】

VVCv2 スライスヘッダー

【表 1 8】

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(!sh_ts_residual_coding_disabled_flag && sps_ts_residual_coding_rice_present_in_sh_flag)	
sh_ts_residual_coding_rice_idx_minus1	u(3)
if(sps_reverse_last_sig_coeff_enabled_flag)	
sh_reverse_last_sig_coeff_flag	u(1)
if(pps_slice_header_extension_present_flag) {	
sh_slice_header_extension_length	ue(v)

10

【0 1 8 5】

図 5 は、本開示のいくつかの実施例に係る、ビデオ復号プロセス 5 0 0 の例を示す。1 つ又は複数のコンピューティング機器は、適切なプログラムコードを実行することによって、図 5 に示す動作を実現する。例えば、ビデオデコーダ 2 0 0 を実現するコンピューティング機器は、プログラムコードを実行することによって、図 5 に示す動作を実現することができる。当該コンピューティング機器は、例えば、エントロピ復号モジュール 2 1 6、逆量子化モジュール 2 1 8、逆変換モジュール 2 1 9、ループ内フィルタモジュール 2 2 0、インター予測モジュール 2 2 4、及びイントラ予測モジュール 2 2 6 を備える。説明のために、図に示されるいくつかの例を参照して、プロセス 5 0 0 について説明する。ただし、他の実現方式も可能である。

20

【0 1 8 6】

ステップ 5 0 2 において、プロセス 5 0 0 は、ビデオ信号（例えば、符号化済みのビデオ 2 0 2 など）のビットストリームにアクセスすることを含む。ステップ 5 0 4 において、プロセス 5 0 0 は、ビデオのビットストリームから共通制約情報（G C I）フラグを抽出することを含む。上述したように、当該バイナリフラグ `g c i _ p r e s e n t _ f l a g` は、G C I 構文要素が存在するか否かを規定するために使用される。`g c i _ p r e s e n t _ f l a g` が 1 に等しい場合、G C I 構文要素が `g e n e r a l _ c o n s t r a i n t s _ i n f o ( )` 構文構造に存在し、G C I 構文要素は、追加符号化ツールにかける制約を示すために使用されることを規定する。`g c i _ p r e s e n t _ f l a g` が 0 に等しい場合、G C I 構文要素が存在せず、ビデオに共通制約をかけないことを規定する。エンコーダに応じて、ビデオのネットワークパケット、ビデオのビデオパラメータセット、又はビデオのシーケンスパラメータセットから G C I フラグを抽出することができる。

30

【0 1 8 7】

ステップ 5 0 6 において、プロセス 5 0 0 は、G C I フラグの値に基づいて、ビデオに 1 つ又は複数の共通制約をかけているか否かを決定することを含む。かけている場合（即ち、G C I フラグが 1 である）、ステップ 5 0 8 において、プロセス 5 0 0 は、ビデオのビットストリームから、ビデオのビットストリームに含まれる追加ビットの数を表す値 M を抽出することを含む。これらの追加ビットは、フラグビットを含み、該フラグビットは、ビデオに対して各追加符号化ツールに制約がかけられていることをそれぞれ示す。

40

【0 1 8 8】

ステップ 5 1 0 において、プロセス 5 0 0 は、M が 5 より大きいのか否かを決定することを含む。そうである場合、ステップ 5 1 2 において、プロセス 5 0 0 は、ビットストリームから 6 つのフラグビットを抽出し、これら 6 つのフラグビットは、6 つの追加符号化ツールのそれぞれの制約を示すフラグを表す。これら 6 つのフラグは、ビデオのピクチャが

50

イントラランダムアクセスポイント (IRAP) ピクチャ又は漸次復号リフレッシュ (GDR) ピクチャであると制限されることを示すフラグ `gci_all_rap_pictures_constraint_flag` と、拡張変換精度を制約しているか否かを示すフラグ `gci_no_extended_precision_processing_constraint_flag` と、明示的なライスパラメータ伝送を制約しているか否かを示すフラグ `gci_no_ts_residual_coding_rice_constraint_flag` と、ビデオの量子化残差を二値化するために使用される代替ライスパラメータ導出を示すフラグ `gci_no_rrc_rice_extension_constraint_flag` と、以前の変換ユニットに基づく二値化のためのライスパラメータ導出を初期化するか否かを示すフラグ `gci_no_persistent_rice_adaptation_constraint_flag` と、TUにおける最後の非ゼロレベルの位置を復号するときに、OlsInScope内のピクチャに制約をかけるか否かを示すフラグ `gci_no_reverse_last_sig_coeff_constraint_flag` と、を含む。

10

【0189】

Mが6より大きい場合、ステップ513において、プロセス500は、ビットストリームから残りのM-6ビットを抽出して破棄することを含む。つまり、ビデオの復号は、このM-6ビットに依存せずに実行される。

【0190】

ステップ514において、プロセス500は、6つのフラグによって6つの追加符号化ツールに対して示される制約に基づいて、ビデオのビットストリームの残りの部分をピクチャに復号することを含む。例えば、フラグ `gci_all_rap_pictures_constraint_flag` が1である場合、デコーダは、1つ又は複数の出力レイヤセット内のすべてのピクチャが `ph_recovery_poc_cnt` が0に等しいGDRピクチャ、又はIRAPピクチャであると決定し、1つ又は複数の出力レイヤセット内のGDRピクチャ又はIRAPピクチャを復号することができる。フラグ `gci_no_extended_precision_processing_constraint_flag` が1である場合、デコーダは、拡張変換精度が制約されていると決定し、拡張ダイナミックレンジを使用しないために、OlsInScope内のピクチャの `sps_extended_precision_flag` を0に設定することによって、ビデオを復号することができる。フラグ `gci_no_ts_residual_coding_rice_constraint_flag` が1である場合、デコーダは、明示的なライスパラメータ伝送が制約されていると決定し、OlsInScope内のピクチャの代替ライスパラメータ伝送を無効にすることによって、ビデオのビットストリームの残りの部分を復号することができる。フラグ `gci_no_rrc_rice_extension_constraint_flag` が1である場合、デコーダは、ビデオの量子化残差を二値化するために使用される代替ライスパラメータ導出が制約されていると決定し、OlsInScope内のピクチャの代替ライスパラメータ伝送を無効にすることによって、ビデオのビットストリームの残りの部分を復号することができる。フラグ `gci_no_persistent_rice_adaptation_constraint_flag` が1である場合、デコーダは、以前の変換ユニット状態に基づく二値化のためのライスパラメータ導出の初期化が制約されていると決定し、OlsInScope内のピクチャに対して以前の変換ユニット状態に基づいてライスパラメータを初期化せずに、ビデオのビットストリームの残りの部分を復号することができる。フラグ `gci_no_reverse_last_sig_coeff_constraint_flag` が1である場合、デコーダは、OlsInScope内のすべてのピクチャの `sps_reverse_last_sig_coeff_enabled_flag` が0に等しく、現在スライスの各変換ブロックについて、最後の重要係数の座標が左上隅 (0, 0) に対して符号化されると決定し、復号された最後の重要係数の座標を、現在スライスの各変換ブロックの左上隅 (0, 0) に対するものであると解釈し、ビデオのビットストリーム

20

30

40

50

の残りの部分を復号することができる。

【0191】

ステップ510において、Mが5以下であると決定された場合、ステップ518において、プロセス500は、ビットストリームからMビットを抽出して破棄することを含む。つまり、ビデオの復号は、このMビットに依存せずに行われる。ステップ520において、デコーダは、6つの追加符号化ツールに制約をかけずに、ビデオを復号する。ステップ506において、GCIフラグがビデオに共通制約をかけないことを示すことを決定した場合（即ち、GCIフラグが0である）、プロセス500は、共通制約なしで、ビデオをピクチャに復号することを含む。いくつかの例では、図2に関して上記で説明したプロセスに従って、復号を実行する。復号済みのビデオは、表示のために出力されることができる。

10

【0192】

図6は、本開示のいくつかの実施例に係る、ビデオ復号プロセス600の別の例を示す。1つ又は複数のコンピューティング機器は、適切なプログラムコードを実行することによって、図6に示す動作を実現する。例えば、ビデオデコーダ200を実現するコンピューティング機器は、プログラムコードを実行することによって、図6に示す動作を実現することができる。当該コンピューティング機器は、例えば、エントリピ復号モジュール216、逆量子化モジュール218、逆変換モジュール219、ループ内フィルタモジュール220、インター予測モジュール224、及びイントラ予測モジュール226を備える。説明のために、図に示されるいくつかの例を参照して、プロセス600について説明する。ただし、他の実現方式も可能である。

20

【0193】

ステップ602において、プロセス600は、ビデオ信号（例えば、符号化済みのビデオ202など）のビットストリームにアクセスすることを含む。ステップ604において、プロセス600は、ビデオのビットストリームから共通制約情報（GCI）フラグを抽出することを含む。上述したように、当該バイナリフラグ `gci_present_flag` は、GCI構文要素が存在するか否かを規定するために使用される。`gci_present_flag` が1に等しい場合、GCI構文要素が `general_constraint` 構文構造に存在し、GCI構文要素は、追加符号化ツールにかける制約を示すために使用されることを規定する。`gci_present_flag` が0に等しい場合、GCI構文要素が存在せず、ビデオに共通制約をかけないことを規定する。エンコーダに応じで、ビデオのネットワークパケット、ビデオのビデオパラメータセット、又はビデオのシーケンスパラメータセットからGCIフラグを抽出することができる。

30

【0194】

ステップ606において、プロセス600は、GCIフラグの値に基づいて、ビデオに1つ又は複数の共通制約をかけているか否かを決定することを含む。かけている場合（即ち、GCIフラグが1である）、ステップ608において、プロセス600は、ビデオのビットストリームから、ビデオのビットストリームに含まれる追加ビットの数を表す値Mを抽出することを含む。これらの追加ビットは、フラグビットを含み、該フラグビットは、ビデオに対して各追加符号化ツールに制約がかけられていることをそれぞれ示す。

40

【0195】

ステップ610において、プロセス600は、Mが6より大きいかなんかを決定することを含む。そうである場合、ステップ612において、プロセス600は、ビットストリームから6つのフラグビットを抽出し、これら6つのフラグビットは、6つの追加符号化ツールのそれぞれの制約を示すフラグを表す。これら6つのフラグは、ビデオのピクチャがイントラランダムアクセスポイント（IRAP）ピクチャ又は漸次復号リフレッシュ（GDR）ピクチャであると制限されることを示すフラグ `gci_all_rap_pictures_constraint_flag` と、拡張変換精度を制約しているかなんかを示すフラグ `gci_no_extended_precision_processing`

50

`_constraint_flag`と、明示的なライスパラメータ伝送を制約しているか否かを示すフラグ`gci_no_ts_residual_coding_rice_constraint_flag`と、ビデオの量子化残差を二値化するために使用される代替ライスパラメータ導出を示すフラグ`gci_no_rrc_rice_extension_constraint_flag`と、以前の変換ユニットに基づく二値化のためのライスパラメータ導出を初期化するか否かを示すフラグ`gci_no_persistent_rice_adaptation_constraint_flag`と、TUにおける最後の非ゼロレベルの位置を復号するときに、`OlsInScope`内のピクチャに制約をかけるか否かを示すフラグ`gci_no_reverse_last_sig_coeff_constraint_flag`と、を含む。

10

【0196】

ステップ614において、プロセス600は、ビットストリームから6つのフラグビットの後ろのM-6ビットを抽出し、抽出されたM-6ビットを破棄することを含む。つまり、ビデオの復号は、このM-6ビットに依存せずに実行される。ステップ616において、プロセス600は、6つのフラグによって6つの追加符号化ツールに対して示される制約に基づいて、ビデオのビットストリームの残りの部分をピクチャに復号することを含む。例えば、フラグ`gci_all_rap_pictures_constraint_flag`が1である場合、デコーダは、1つ又は複数の出力レイヤセット内のすべてのピクチャが`ph_recovery_poc_cnt`が0に等しいGDRピクチャ、又はIRAPピクチャであると決定し、1つ又は複数の出力レイヤセット内のGDRピクチャ又はIRAPピクチャを復号することができる。フラグ`gci_no_extended_precision_processing_constraint_flag`が1である場合、デコーダは、拡張変換精度が制約されていると決定し、拡張ダイナミックレンジを使用しないために、`OlsInScope`内のピクチャの`sps_extended_precision_flag`を0に設定することによって、ビデオを復号することができる。フラグ`gci_no_ts_residual_coding_rice_constraint_flag`が1である場合、デコーダは、明示的なライスパラメータ伝送が制約されていると決定し、`OlsInScope`内のピクチャの代替ライスパラメータ伝送を無効にすることによって、ビデオのビットストリームの残りの部分を復号することができる。フラグ`gci_no_rrc_rice_extension_constraint_flag`が1である場合、デコーダは、ビデオの量子化残差を二値化するために使用される代替ライスパラメータ導出が制約されていると決定し、`OlsInScope`内のピクチャの代替ライスパラメータ伝送を無効にすることによって、ビデオのビットストリームの残りの部分を復号することができる。フラグ`gci_no_persistent_rice_adaptation_constraint_flag`が1である場合、デコーダは、以前の変換ユニット状態に基づく二値化のためのライスパラメータ導出の初期化が制約されていると決定し、`OlsInScope`内のピクチャに対して以前の変換ユニット状態に基づいてライスパラメータを初期化せずに、ビデオのビットストリームの残りの部分を復号することができる。フラグ`gci_no_reverse_last_sig_coeff_constraint_flag`が1である場合、デコーダは、`OlsInScope`内のすべてのピクチャの`sps_reverse_last_sig_coeff_enabled_flag`が0に等しく、現在スライスの各変換ブロックについて、最後の重要係数の座標が左上隅(0, 0)に対して符号化されると決定し、復号された最後の重要係数の座標を、現在スライスの各変換ブロックの左上隅(0, 0)に対するものであると解釈し、ビデオのビットストリームの残りの部分を復号することができる。

20

30

40

【0197】

ステップ610において、Mが6以下であると決定された場合、Mは、0又は6である。Mが6である場合、ステップ620において、プロセス600は、前述のように、6つのフラグビットを抽出することを含む。ステップ622において、デコーダは、6つの追

50

加符号化ツールに制約をかけることによって、抽出されたフラグビットに基づいてビデオを復号する。Mが0である場合、フラグビットを抽出しない。ステップ624において、デコーダは、フラグビットが抽出されていないため、6つの追加符号化ツールに制約をかけずに、ビデオを復号する。ステップ606において、GCIフラグがビデオに共通制約をかけないことを示すことを決定した場合（即ち、GCIフラグが0である）、ステップ618において、プロセス600は、共通制約なしで、ビデオをピクチャに復号することを含む。いくつかの例では、図2に関して上記で説明したプロセスに従って、復号を実行する。復号済みのビデオは、表示のために出力されることができる。

【0198】

図7は、本開示のいくつかの実施例に係る、ビデオ復号プロセス700の別の例を示す。1つ又は複数のコンピューティング機器は、適切なプログラムコードを実行することによって、図7に示す動作を実現する。例えば、ビデオデコーダ200を実現するコンピューティング機器は、プログラムコードを実行することによって、図7に示す動作を実現することができる。当該コンピューティング機器は、例えば、エンロピ復号モジュール216、逆量子化モジュール218、逆変換モジュール219、ループ内フィルタモジュール220、インター予測モジュール224、及びイントラ予測モジュール226を備える。説明のために、図に示されるいくつかの例を参照して、プロセス700について説明する。ただし、他の実現方式も可能である。

【0199】

ステップ702において、プロセス700は、ビデオ信号（例えば、符号化済みのビデオ202など）のビットストリームにアクセスすることを含む。ステップ704において、プロセス700は、ビデオのビットストリームから共通制約情報（GCI）フラグを抽出することを含む。上述したように、当該バイナリフラグ `gci_present_flag` は、GCI構文要素が存在するか否かを規定するために使用される。`gci_present_flag` が1に等しい場合、GCI構文要素が `general_constraints_info()` 構文構造に存在し、GCI構文要素は、追加符号化ツールにかける制約を示すために使用されることを規定する。`gci_present_flag` が0に等しい場合、GCI構文要素が存在せず、ビデオに共通制約をかけないことを規定する。エンコーダに依拠して、ビデオのネットワークパケット、ビデオのビデオパラメータセット、又はビデオのシーケンスパラメータセットからGCIフラグを抽出することができる。

【0200】

ステップ706において、プロセス700は、GCIフラグの値に基づいて、ビデオに1つ又は複数の共通制約をかけているか否かを決定することを含む。かけている場合（即ち、GCIフラグが1である）、ステップ708において、プロセス700は、ビデオのビットストリームから、ビデオのビットストリームに含まれる追加ビットの数を表す値Mを抽出することを含む。これらの追加ビットは、フラグビットを含み、該フラグビットは、ビデオに対して各追加符号化ツールに制約がかけられていることをそれぞれ示す。

【0201】

ステップ710において、プロセス700は、Mが5より大きいのか否かを決定することを含む。そうでない場合、ステップ712において、プロセス700は、ビットストリームからMビットを抽出し、このMビットを破棄することを含む。ステップ714において、プロセス700は、このMビットに依存せずに、ビデオのビットストリームの残りの部分をピクチャに復号することを含む。ステップ710において、Mが5以下であると決定された場合、ステップ718において、プロセス700は、上記の6つのフラグビットを抽出することを含む。Mが6より大きい場合、ステップ719において、プロセス700は、ビットストリームから残りのM-6ビットを抽出して破棄することを含む。つまり、ビデオの復号は、このM-6ビットに依存せずに実行される。ステップ720において、上記のように、デコーダは、6つの追加符号化ツールに制約をかけることによって、抽出された6つのフラグビットに基づいてビデオを復号する。

10

20

30

40

50

【0202】

ステップ706において、GCIフラグがビデオに共通制約をかけないことを示すことを決定した場合（即ち、GCIフラグが0である）、プロセス700は、共通制約なしで、ビデオをピクチャに復号することを含む。いくつかの例では、図2に関して上記で説明したプロセスに従って、復号を実行する。復号済みのビデオは、表示のために出力されることができる。

【0203】

共通制約情報の伝送を実現するためのコンピューティングシステムの例

本明細書で説明される動作を実行するために、任意の適切なコンピューティングシステムを使用することができる。例えば、図8は、図1のビデオエンコーダ100又は図2のビデオデコーダ200を実装することができるコンピューティング機器800の例を示す。いくつかの実施例では、コンピューティング機器800は、プロセッサ812を備えることができ、プロセッサ812は、メモリ814に通信可能に結合され、コンピュータ実行可能なプログラムコードを実行し、及び／又はメモリ814に記憶された情報にアクセスする。プロセッサ812は、マイクロプロセッサ、特定用途向け集積回路（「ASIC」）、ステートマシン、又は他の処理機器を含むことができる。プロセッサ812は、複数の処理機器のうちの任意の1つの処理機器を含み得る。このようなプロセッサは、命令を記憶するコンピュータ可読媒体を含んでもよいし、命令を記憶するコンピュータ可読媒体と通信してもよく、命令がプロセッサ812によって実行されるとき、プロセッサに本明細書で説明されたステップを実行させる。

【0204】

メモリ814は、任意の適切な非一時的なコンピュータ可読媒体を含むことができる。コンピュータ可読媒体は、コンピュータ可読命令又はその他のプログラムコードをプロセッサに提供することができる、任意の電子、光学、磁気又はその他の記憶装置を含むことができる。コンピュータ可読媒体の非限定的な例には、磁気ディスク、メモリチップ、ROM、RAM、ASIC、構成されたプロセッサ、光学メモリ、磁気テープ又はその他の磁気ストレージ、又はコンピュータプロセッサが命令を読み取ることができる任意の他の媒体を含む。命令は、コンパイラ及び／又はインタープリタが任意の適切なコンピュータプログラミング言語（例えば、C、C++、C#、Visual Basic、Java（登録商標）、Python、Perl、JavaScript、及びActionScriptなどを含む）で記述されたコードに基づいて生成されたプロセッサ特定命令を含み得る。

【0205】

コンピューティング機器800はさらに、バス816を備えることができる。バス816は、コンピューティング機器800の1つ又は複数のコンポーネントを通信可能に結合することができる。コンピューティング機器800はさらに、入力又は出力機器など、複数の外部又は内部機器を備えることができ、例えば、示されたコンピューティング機器800は、入力／出力（「I/O」）インターフェース818を備え、1つ又は複数の入力機器820から入力を受信するか、又は1つ又は複数の出力機器822に出力を提供することができる。1つ又は複数の入力機器820と1つ又は複数の出力機器822は、I/Oインターフェース818に通信可能に結合されることができる。当該通信結合は、任意の適切な方式（例えば、プリント回路基板を介した接続、ケーブルを介した接続、無線伝送を介した通信など）によって実現されることができる。入力機器820の非限定的な例には、タッチスクリーン（例えば、タッチ領域を撮影するための1つ又は複数のカメラ、又はタッチによって引き起こされる圧力変化を検出するための圧力センサ）、マウス、キーボード、又はコンピューティング機器のユーザの物理的な動きに応答して入力イベントを生成するための任意の他の機器を含む。出力機器822の非限定的な例には、LCDスクリーン、外部モニタ、スピーカ、又はコンピューティング機器によって生成される出力を表示又は他の方式で提示するための任意の他の機器を含む。

【0206】

コンピューティング機器 800 は、プログラムコードを実行することができ、当該プログラムコードは、プロセッサ 812 に、図 1～図 7 に関して上記で説明された 1 つ又は複数のステップを実行させる。プログラムコードは、ビデオエンコーダ 100 又はビデオデコーダ 200 を含むことができる。プロプログラムコードは、メモリ 814 又は任意の適切なコンピュータ可読媒体に常駐することができ、プロセッサ 812 又は任意の他の適切なプロセッサによって実行されることができる。

【0207】

コンピューティング機器 800 はさらに、少なくとも 1 つのネットワークインターフェース機器 824 を備えることができる。ネットワークインターフェース機器 824 は、1 つ又は複数のデータネットワーク 828 への有線又は無線データ接続を確立するのに適した任意の機器又は機器グループを含むことができる。ネットワークインターフェース機器 824 の非限定的な例には、イーサネットワークアダプタ、無線モデム及び／又は類似する機器などを含む。コンピューティング機器 800 は、ネットワークインターフェース機器 824 を介してメッセージを電子信号又は光学信号として伝送することができる。

10

【0208】

一般的な考慮事項

本明細書には、主張された主題の完全な理解を提供するために、多数の詳細が記載されている。しかし、当業者であれば理解できるように、これらの詳細なしにも、主張された主題を実施することができる。他の例では、主張された主題を曖昧にしないように、当業者に既知の方法、装置又はシステムは、詳細に説明されていない。

20

【0209】

特に明記しない限り、本明細書では、「処理」、「コンピューティング (computing)」、「計算 (calculating)」、「決定」、「識別」などの用語を使用して説明するのは、コンピューティング機器 (例えば、1 つ又は複数のコンピュータ又は類似の電子コンピューティング機器) の動作又はプロセスを指し、これらの機器は、コンピューティングプラットフォームのメモリ、レジスタ又は他の情報記憶装置、伝送機器又は表示機器内の物理的な電子又は磁気量として表されるデータを操作又は変換する。

【0210】

本明細書で説明するシステムは、任意の特定のハードウェアアーキテクチャ又は構成に限定されない。コンピューティング機器は、1 つ又は複数の入力に条件付けられた結果を提供するコンポーネントの任意の適切な配置を含んでもよい。適切なコンピューティング機器は、記憶されたソフトウェアにアクセスするマイクロプロセッサに基づく多用途コンピュータシステムを含み、当該ソフトウェアは、コンピューティングシステムを汎用コンピューティング装置から本明細書の主題の 1 つ又は複数の実施例を実現する専用コンピューティング装置にプログラミング又は構成させる。任意の適切なプログラミング、スクリプト、又は他の種類の言語又は言語の組み合わせは、コンピューティング機器をプログラミング又は構成するように構成されるソフトウェアで本明細書に含まれる教示を実現するために使用されることができる。

30

【0211】

本明細書で開示された方法の実施例は、このようなコンピューティング機器の動作で実行されることができる。上記の例で示したステップの順序は変更でき、例えば、ステップの順序を変更したり、結合したり、サブステップに分割したりすることができる。いくつかのステップ又はプロセスは、並行して実行できる。

40

【0212】

本明細書で使用される「…に適用される」又は「…に構成される」は、開放的かつ包括的な言語であり、追加のタスク又はステップを実行するように適用又は構成された機器を排除しない。さらに、「…に基づいて」の使用は、開放的かつ包括的であり、これは、1 つ又は複数の列挙された条件又は値「に基づく」プロセス、ステップ、計算又は他の動作は、実際には、列挙されたものを超える追加の条件又は値に基づいている可能性があるためである。本明細書に含まれる見出し、リスト及び番号付けは、説明を容易にするためだ

50

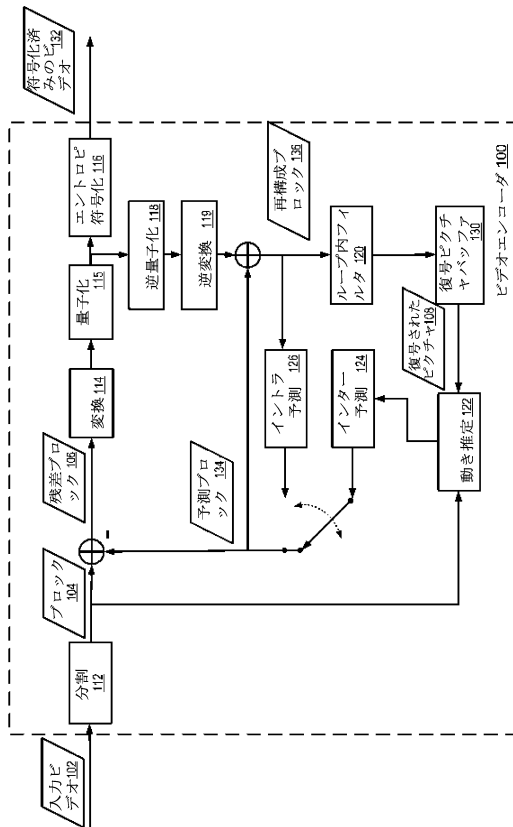
けのものであり、限定することを意図しない。

【0213】

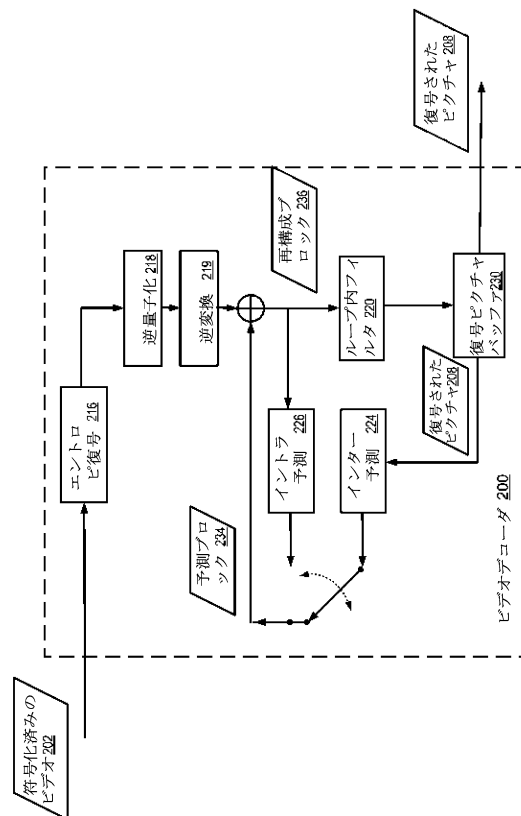
本明細書の主題については、その具体的な実施例に関して詳細に説明されているが、当業者であれば、前述の内容を理解した後、このような実施例に容易に変更、変形、及び同等のものをもたらすことができることを理解されたい。したがって、理解すべきこととして、本開示は、限定ではなく例示の目的で提示されており、当業者にとって容易になし得る、本明細書の主題に対するこのような変更、変形及び／又は追加を含むことを排除しない。

【図面】

【図 1】



【図 2】



10

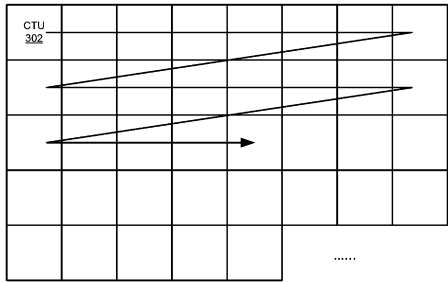
20

30

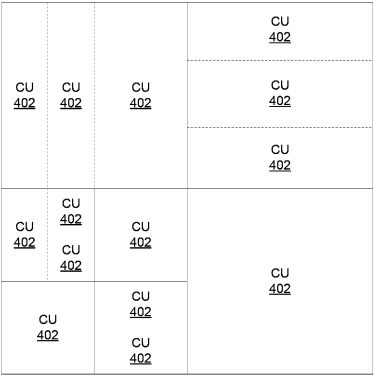
40

50

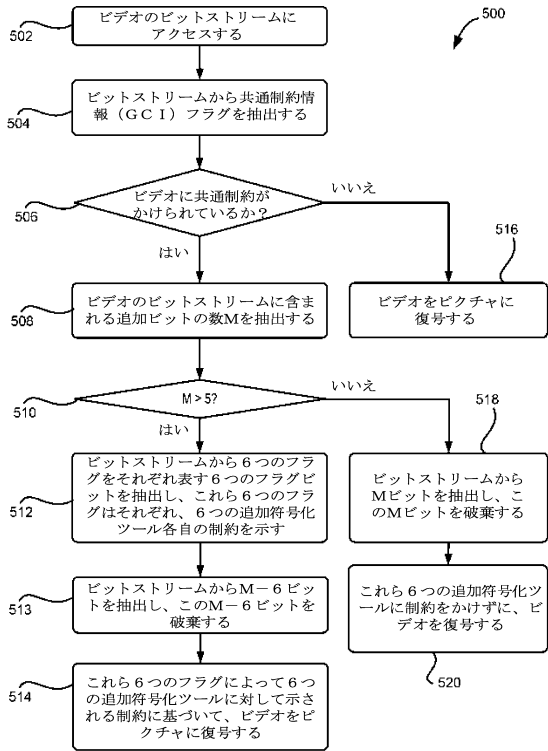
【図 3】



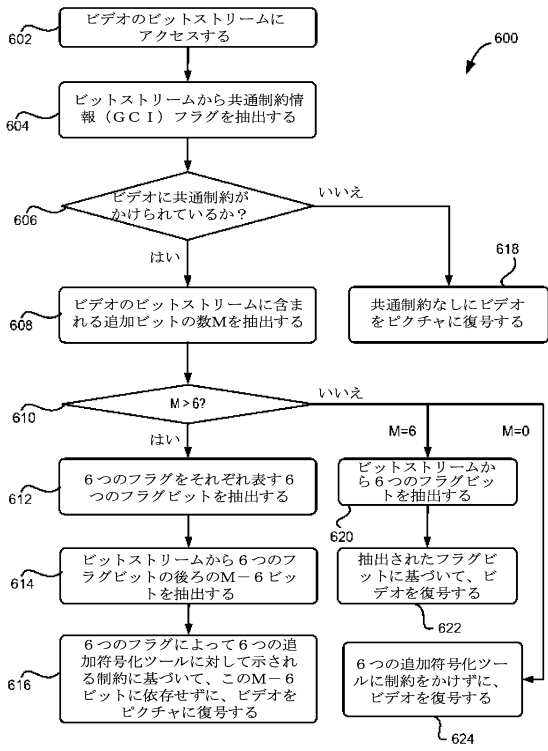
【図 4】



【図 5】



【図 6】



10

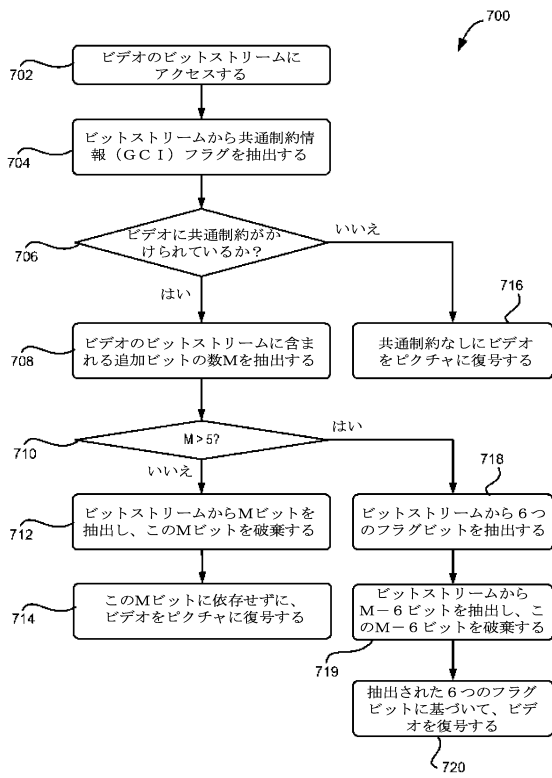
20

30

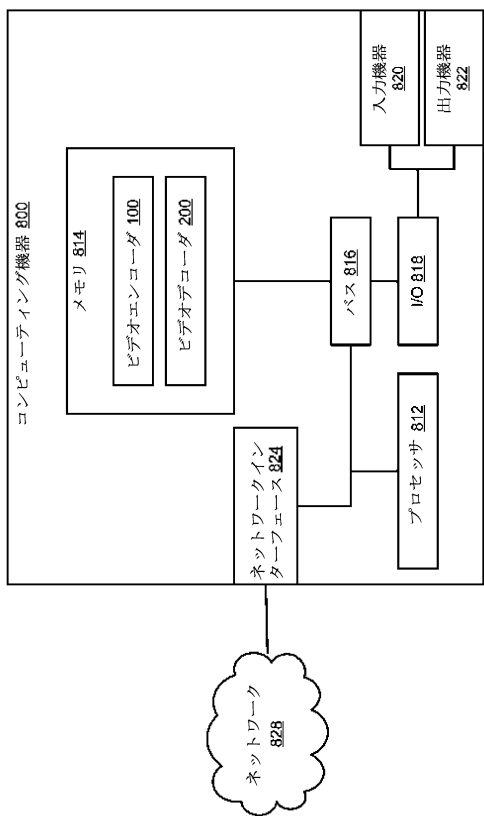
40

50

【図 7】



【図 8】



10

20

30

40

50

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2022/079490

A. CLASSIFICATION OF SUBJECT MATTER IPC(8) - INV. - H04N 19/70; H04N 19/124 (2022.01) ADD. - H04N 19/136 (2022.01) CPC - INV. - H04N 19/70; H04N 19/124 (2022.08) ADD. - H04N 19/136; H04N 19/119; H04N 19/172; H04N 19/176; H04N 19/184 (2022.08) According to International Patent Classification (IPC) or to both national classification and IPC		
B. FIELDS SEARCHED		
Minimum documentation searched (classification system followed by classification symbols) See Search History document		
Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched See Search History document		
Electronic database consulted during the international search (name of database and, where practicable, search terms used) See Search History document		
C. DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2021/0314823 A1 (QUALCOMM INCORPORATED) 07 October 2021 (07.10.2021) entire document	1-20
A	WO 2021/216736 A1 (DOLBY LABORATORIES LICENSING CORPORATION) 28 October 2021 (28.10.2021) entire document	1-20
A	US 2021/0136369 A1 (SK TELECOM CO. LTD.) 06 May 2021 (06.05.2021) entire document	1-20
A	WO 2021/252396 A1 (BYTEDANCE INC.) 16 December 2021 (16.12.2021) entire document	1-20
A	WO 2021/202178 A1 (BEIJING DAJIA INTERNET INFORMATION TECHNOLOGY CO. LTD.) 07 October 2021 (07.10.2021) entire document	1-20
A	Wenger "H.264/AVC Over IP"; Published In: IEEE Transactions on Circuits and Systems for Video Technology (Volume: 13, Issue: 7, July 2003); Date of Publication: July 2003; retrieved on [23.12.2022]; retrieved from the Internet <URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1218197 > entire document	1-20
☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.		
* Special categories of cited documents: "A" document defining the general state of the art which is not considered to be of particular relevance "D" document cited by the applicant in the international application "E" earlier application or patent but published on or after the international filing date "L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) "O" document referring to an oral disclosure, use, exhibition or other means "P" document published prior to the international filing date but later than the priority date claimed "T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention "X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone "Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art "&" document member of the same patent family		
Date of the actual completion of the international search 22 December 2022		Date of mailing of the international search report FEB 02 2023
Name and mailing address of the ISA/ Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, VA 22313-1450 Facsimile No. 571-273-8300		Authorized officer Taina Matos Telephone No. PCT Helpdesk: 571-272-4300

Form PCT/ISA/210 (second sheet) (July 2022)

フロントページの続き

(32)優先日 令和4年1月13日(2022.1.13)

(33)優先権主張国・地域又は機関
米国(US)

(81)指定国・地域 AP(BW,CV,GH,GM,KE,LR,LS,MW,MZ,NA,RW,SD,SL,ST,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,RU,TJ,TM),EP(AL,AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,HR,HU,IE,IS,IT,LT,LU,LV,MC,ME,MK,MT,NL,NO,PL,PT,RO,RS,SE,SI,SK,SM,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,KM,ML,MR,NE,SN,TD,TG),AE,AG,AL,AM,AO,AT,AU,AZ,BA,BB,BG,BH,BN,BR,BW,BY,BZ,CA,CH,CL,CN,CO,CR,CU,CV,CZ,DE,DJ,DK,DM,DO,DZ,EC,EE,EG,ES,FI,GB,GD,GE,GH,GM,GT,HN,HR,HU,ID,IL,IN,IQ,IR,IS,IT,JM,JO,JP,KE,KG,KH,KN,KP,KR,KW,KZ,LA,LC,LK,LR,LS,LU,LY,MA,MD,MG,MK,MN,MW,MX,MY,MZ,NA,NG,NI,NO,NZ,OM,PA,PE,PG,PH,PL,PT,QA,RO,RS,RU,RW,SA,SC,SD,SE,SG,SK,SL,ST,SV,SY,TH,TJ,TM,TN,TR,TT,TZ,UA,UG,US,UZ,VC,VN,WS,ZA,ZM,ZW

(特許庁注：以下のものは登録商標)

1. VISUAL BASIC

2. PYTHON

3. JAVASCRIPT

4. イーサネット

(72)発明者 ガン ジョナサン

アメリカ合衆国 カリフォルニア州 94303 パロ アルト イースト ベイショア ロード 24
79 スイート 110 イノピーク テクノロジー インコーポレイテッド内

(72)発明者 ユー ユエ

アメリカ合衆国 カリフォルニア州 94303 パロ アルト イースト ベイショア ロード 24
79 スイート 110 イノピーク テクノロジー インコーポレイテッド内

(72)発明者 ユー ハオピン

アメリカ合衆国 カリフォルニア州 94303 パロ アルト イースト ベイショア ロード 24
79 スイート 110 イノピーク テクノロジー インコーポレイテッド内

Fターム (参考) 5C159 MA04 MA05 MC11 ME01 PP04 RC11 UA02 UA05 UA16