(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)公表番号
特表2025-504773
(P2025-504773A)
(43)公表日　令和7年2月19日(2025.2.19)

(51)国際特許分類	F I	テーマコード（参考）
H04N 19/70 (2014.01)	H04N 19/70	5C159

審査請求　未請求　予備審査請求　未請求　（全67頁）

(21)出願番号　特願2024-539853(P2024-539853)
(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5日(2024.7.5)
(86)国際出願番号　PCT/US2022/079494
(87)国際公開番号　WO2023/132993
(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 MOBILE TELECOMMUNICATIONS CORP., LTD.
中華人民共和国広東省東莞市長安鎮烏沙海浜路１８号
No. 18 Haibin Road, Wusha, Chang'an, Dongguan, Guangdong 523860 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一般制約情報のシグナリング

(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に基づいてビデオを復号化する。ビデオデコーダは、ビデオのビットストリームにアクセスし、ビデオのビットストリームから一般制約情報（ＧＣＩ）フラグを抽出する。デコーダは、ＧＣＩフラグの値に基づいて、ビデオに１つ又は複数の一般制約が課されていると決定し、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示す値を抽出する。追加ビットは、フラグビットを含み、当該フラグビットは、ビデオに対して制約される対応する追加符号化ツールを示す。当該値が５より大きい場合、デコーダは、ビデオのビットストリームから６つのフラグを抽出し、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対応する制約を示す。デコーダは、これら６つのフラグによって示される６つの追加符号化ツールの制約に基づいて、ビデオのビットストリームを画像に復号化する。
【選択図】図５

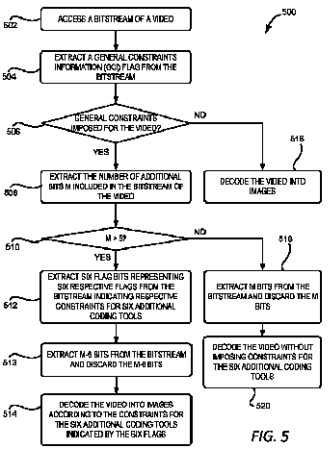


FIG. 5

【特許請求の範囲】
【請求項１】
ビデオ復号化方法であって、
ビデオのビットストリームから一般制約情報（ＧＣＩ）フラグを抽出することと、
前記ＧＣＩフラグの値に基づいて、前記ビデオに１つ又は複数の一般制約が課されていると決定することと、
前記ビデオに前記１つ又は複数の一般制約が課されていると決定したことに応答して、前記ビデオのビットストリームから、前記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示す値を抽出することであって、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対応する追加符号化ツールを示す、ことと、
前記値が５より大きいと決定することと、
前記値が５より大きいと決定したことに応答して、
前記ビットストリームから、６つの対応するフラグを表す６つのフラグビットを抽出することであって、前記６つの対応するフラグは、６つの追加符号化ツールの対応する制約を示す、ことと、
前記６つのフラグによって示される前記６つの追加符号化ツールの制約に少なくとも部分的に基づいて、前記ビデオのビットストリームの残りの部分を画像に復号化することと、を含む、ビデオ復号化方法。
【請求項２】
前記６つのフラグは、
前記ビデオの画像がイントラランダムアクセスポイント（ＩＲＡＰ）画像又は漸次復号リフレッシュ（ＧＤＲ）画像に制限されることを示す第１フラグと、
拡張変換精度が制約され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シグナリングが制約され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いて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化するときに、スコープ内出力レイヤセット（ＯｌｓＩｎＳｃｏｐｅ）内の画像に制約を課すか否かを示す第６フラグと、を含む、
請求項１に記載のビデオ復号化方法。
【請求項３】
前記ビデオ復号化方法は、
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課さ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課されていないことを示し、対応する符号化ツールに制約が課さ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課さ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課さ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課されていないことを示すこと、又は
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課さ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２に記載のビデオ復号化方法。
【請求項４】
前記６つのフラグによって示される前記６つの追加符号化ツールの制約に少なくとも部分的に基づいて、前記ビデオのビットストリームの残りの部分を画像に復号化することは、
前記第１フラグが１であることに基づいて、１つ又は複数の出力レイヤセット内のすべての画像がＩＲＡＰ画像又はｐｈ＿ｒｅｃｏｖｅｒｙ＿ｐｏｃ＿ｃｎｔが０に等しいＧＤＲ画像であると決定し、前記１つ又は複数の出力レイヤセット内のＧＤＲ画像又はＩＲＡＰ画像を復号化すること、
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、スコープ内出力レイヤセット（ＯｌｓＩｎＳｃｏｐｅ）内の画像のｓｐｓ＿ｅｘｔｅｎｄｅｄ＿ｐｒｅｃｉｓｉｏｎ＿ｆｌａｇを０に設定することによって、前記ビデオのビットストリームの残りの部分を復号化すること、
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シグナリングが制約されていると決定し、前記スコープ内出力レイヤセット（ＯｌｓＩｎＳｃｏｐｅ）内の画像の代替ライスパラメータシグナリングを無効にすることによって、前記ビデオのビットストリームの残りの部分を復号化すること、
前記第３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を二値化する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、前記スコープ内出力レイヤセット（ＯｌｓＩｎＳｃｏｐｅ）内の画像の代替ライスパラメータシグナリングを無効にすることによって、前記ビデオのビットストリームの残りの部分を復号化すること、
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、前記スコープ内出力レイヤセット（ＯｌｓＩｎＳｃｏｐｅ）内の画像の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化すること、又は
前記第６フラグが１であることに基づいて、最後の有効係数（ｌａｓｔ　ｓｉｇｎｉｆｉｃａｎｔ　ｃｏｅｆｆｉｃｉｅｎｔ）の座標がスライスの各変換ブロックの左上隅を基準として符号化されると決定し、復号化された前記最後の有効係数の座標を、前記スライスの各変換ブロックの前記左上隅を基準としたものとして解釈することによって、前記ビデオのビットストリームの残りの部分を復号化すること、のうちの１つ又は複数を含む、
請求項２に記載のビデオ復号化方法。
【請求項５】
前記ＧＣＩ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、
請求項１に記載のビデオ復号化方法。
【請求項６】
前記ビデオ復号化方法は、
前記値が５より大きいと決定したことに応答して、抽出された追加ビットを示す値ｎｕｍＡｄｄｉｔｉｏｎａｌＢｉｔｓＵｓｅｄを６に設定することと、
前記ビットストリームからビットセットを抽出することであって、前記ビットセット内のビット数は、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓ－ｎｕｍＡｄｄｉｔｉｏｎａｌＢｉｔｓＵｓｅｄに等しく、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓは、前記値を表すことと、
前記ビットセットから独立して、前記ビデオのビットストリームの残りの部分を画像に復号化することと、をさらに含む、
請求項１に記載のビデオ復号化方法。
【請求項７】
前記ビデオ復号化方法は、

前記値が５以下であると決定したことに応答して、抽出された追加ビットを示す値ｎｕｍＡｄｄｉｔｉｏｎａｌＢｉｔｓＵｓｅｄを０に設定することと、
前記ビットストリームからビットセットを抽出することであって、前記ビットセット内のビット数は、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓ－ｎｕｍＡｄｄｉｔｉｏｎａｌＢｉｔｓＵｓｅｄに等しく、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓは、前記値を表すことと、
抽出されたビットセットから独立して、前記ビデオのビットストリームの残りの部分を画像に復号化することと、をさらに含む、
請求項１に記載のビデオ復号化方法。
【請求項８】
プログラムコードが記憶され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一般制約情報（ＧＣＩ）フラグを抽出することと、
前記ＧＣＩフラグの値に基づいて、前記ビデオに１つ又は複数の一般制約が課されていると決定することと、
前記ビデオに前記１つ又は複数の一般制約が課されていると決定したことに応答して、前記ビデオのビットストリームから、前記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示す値を抽出することであって、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対応する追加符号化ツールを示す、ことと、
前記値が５より大きいと決定することと、
前記値が５より大きいと決定したことに応答して、
前記ビットストリームから、６つの対応するフラグを表す６つのフラグビットを抽出することであって、前記６つの対応するフラグは、６つの追加符号化ツールの対応する制約を示す、ことと、
前記６つのフラグによって示される前記６つの追加符号化ツールの制約に少なくとも部分的に基づいて、前記ビデオのビットストリームの残りの部分を画像に復号化することと、を含む、非一時的なコンピュータ可読媒体。
【請求項９】
前記６つのフラグは、
前記ビデオの画像がイントラランダムアクセスポイント（ＩＲＡＰ）画像又は漸次復号リフレッシュ（ＧＤＲ）画像に制限されることを示す第１フラグと、
拡張変換精度が制約され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シグナリングが制約され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いて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化するときに、スコープ内出力レイヤセット（ＯｌｓＩｎＳｃｏｐｅ）内の画像に制約を課すか否かを示す第６フラグと、を含む、
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課さ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課されていないことを示し、対応する符号化ツールに制約が課さ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課さ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課さ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課されていないことを示すこと、又は
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課さ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９に記載の非一時的なコンピュータ可読媒体。

【請求項１１】
前記６つのフラグによって示される前記６つの追加符号化ツールの制約に少なくとも部分的に基づいて、前記ビデオのビットストリームの残りの部分を画像に復号化することは、
前記第１フラグが１であることに基づいて、１つ又は複数の出力レイヤセット内のすべての画像がＩＲＡＰ画像又はｐｈ＿ｒｅｃｏｖｅｒｙ＿ｐｏｃ＿ｃｎｔが０に等しいＧＤＲ画像であると決定し、前記１つ又は複数の出力レイヤセット内のＧＤＲ画像又はＩＲＡＰ画像を復号化すること、
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、スコープ内出力レイヤセット（ＯｌｓＩｎＳｃｏｐｅ）内の画像のｓｐｓ＿ｅｘｔｅｎｄｅｄ＿ｐｒｅｃｉｓｉｏｎ＿ｆｌａｇを０に設定して、前記ビデオのビットストリームの残りの部分を復号化すること、
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シグナリングが制約されていると決定し、前記スコープ内出力レイヤセット（ＯｌｓＩｎＳｃｏｐｅ）内の画像の代替ライスパラメータシグナリングを無効にすることによって、前記ビデオのビットストリームの残りの部分を復号化すること、
前記第３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を二値化する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、前記スコープ内出力レイヤセット（ＯｌｓＩｎＳｃｏｐｅ）内の画像の代替ライスパラメータシグナリングを無効にすることによって、前記ビデオのビットストリームの残りの部分を復号化すること、
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、前記スコープ内出力レイヤセット（ＯｌｓＩｎＳｃｏｐｅ）内の画像の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化すること、又は
前記第６フラグが１であることに基づいて、最後の有効係数の座標がスライスの各変換ブロックの左上隅を基準として符号化されると決定し、復号化された前記最後の有効係数の座標を、前記スライスの各変換ブロックの前記左上隅を基準としたものとして解釈することによって、前記ビデオのビットストリームの残りの部分を復号化すること、のうちの１つ又は複数を含む、
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１２】
前記ＧＣＩ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、
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１３】
前記動作は、
前記値が５より大きいと決定したことに応答して、抽出された追加ビットを示す値ｎｕｍＡｄｄｉｔｉｏｎａｌＢｉｔｓＵｓｅｄを６に設定することと、
前記ビットストリームからビットセットを抽出することであって、前記ビットセット内

のビット数は、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓ－ｎｕｍＡｄｄｉｔｉｏｎａｌＢｉｔｓＵｓｅｄに等しく、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓは、前記値を表すことと、

前記ビットセットから独立して、前記ビデオのビットストリームの残りの部分を画像に復号化することと、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】

前記動作は、

前記値が５以下であると決定したことに応答して、抽出された追加ビットを示す値ｎｕｍＡｄｄｉｔｉｏｎａｌＢｉｔｓＵｓｅｄを０に設定することと、

前記ビットストリームからビットセットを抽出することであって、前記ビットセット内のビット数は、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓ－ｎｕｍＡｄｄｉｔｉｏｎａｌＢｉｔｓＵｓｅｄに等しく、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓは、前記値を表すことと、

抽出されたビットセットから独立して、前記ビデオのビットストリームの残りの部分を画像に復号化することと、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１５】

システムであって、

処理機器と、

前記処理機器と通信可能に結号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一般制約情報（ＧＣＩ）フラグを抽出することと、

前記ＧＣＩフラグの値に基づいて、前記ビデオに１つ又は複数の一般制約が課されていると決定することと、

前記ビデオに前記１つ又は複数の一般制約が課されていると決定したことに応答して、前記ビデオのビットストリームから、前記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示す値を抽出することであって、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対応する追加符号化ツールを示す、ことと、

前記値が５より大きいと決定することと、

前記値が５より大きいと決定したことに応答して、

前記ビットストリームから、６つの対応するフラグを表す６つのフラグビットを抽出することであって、前記６つの対応するフラグは、６つの追加符号化ツールの対応する制約を示す、ことと、

前記６つのフラグによって示される前記６つの追加符号化ツールの制約に少なくとも部分的に基づいて、前記ビデオのビットストリームの残りの部分を画像に復号化することと、を含む、システム。

【請求項１６】

前記６つのフラグは、

前記ビデオの画像がイントラランダムアクセスポイント（ＩＲＡＰ）画像又は漸次復号リフレッシュ（ＧＤＲ）画像に制限されることを示す第１フラグと、

拡張変換精度が制約され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シグナリングが制約され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いて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化するときに、スコープ内出力

レイヤセット（ＯｌｓＩｎＳｃｏｐｅ）内の画像に制約を課すか否かを示す第６フラグと、を含む、
請求項１５に記載のシステム。
【請求項１７】
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課さ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課されていないことを示し、対応する符号化ツールに制約が課さ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課さ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課さ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課されていないことを示すこと、又は
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課さ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１６に記載のシステム。
【請求項１８】
前記６つのフラグによって示される前記６つの追加符号化ツールの制約に少なくとも部分的に基づいて、前記ビデオのビットストリームの残りの部分を画像に復号化することは、
前記第１フラグが１であることに基づいて、１つ又は複数の出力レイヤセット内のすべての画像がＩＲＡＰ画像又はｐｈ＿ｒｅｃｏｖｅｒｙ＿ｐｏｃ＿ｃｎｔが０に等しいＧＤＲ画像であると決定し、前記１つ又は複数の出力レイヤセット内のＧＤＲ画像又はＩＲＡＰ画像を復号化すること、
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、スコープ内出力レイヤセット（ＯｌｓＩｎＳｃｏｐｅ）内の画像のｓｐｓ＿ｅｘｔｅｎｄｅｄ＿ｐｒｅｃｉｓｉｏｎ＿ｆｌａｇを０に設定して、前記ビデオの前記ビットストリームの前記残りの部分を復号化すること、
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シグナリングが制約されていると決定し、前記スコープ内出力レイヤセット（ＯｌｓＩｎＳｃｏｐｅ）内の画像の代替ライスパラメータシグナリングを無効にすることによって、前記ビデオのビットストリームの残りの部分を復号化すること、
前記第３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を二値化する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、前記スコープ内出力レイヤセット（ＯｌｓＩｎＳｃｏｐｅ）内の画像の代替ライスパラメータシグナリングを無効にすることによって、前記ビデオのビットストリームの残りの部分を復号化すること、
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、前記スコープ内出力レイヤセット（ＯｌｓＩｎＳｃｏｐｅ）内の画像の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化すること、又は
前記第６フラグが１であることに基づいて、最後の有効係数の座標がスライスの各変換ブロックの左上隅を基準として符号化されると決定し、復号化された前記最後の有効係数の座標を、前記スライスの各変換ブロックの前記左上隅を基準としたものとして解釈することによって、前記ビデオのビットストリームの残りの部分を復号化すること、のうちの

１つ又は複数を含む、
請求項１６に記載のシステム。
【請求項１９】
前記ＧＣＩ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、
請求項１５に記載のシステム。
【請求項２０】
前記動作は、
前記値が５より大きいと決定したことに応答して、抽出された追加ビットを示す値ｎｕｍＡｄｄｉｔｉｏｎａｌＢｉｔｓＵｓｅｄを６に設定することと、
前記ビットストリームからビットセットを抽出することであって、前記ビットセット内のビット数は、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓ－ｎｕｍＡｄｄｉｔｉｏｎａｌＢｉｔｓＵｓｅｄに等しく、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓは、前記値を表すことと、
前記ビットセットから独立して、前記ビデオのビットストリームの残りの部分を画像に復号化することと、をさらに含む、
請求項１５に記載のシステム。
【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、及び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、これらの米国仮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】
本発明は、総体的に、ビデオ処理に関する。具体的には、本発明は、ビデオ符号化のための一般制約情報のシグナリングと初期化に関する。
【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、ビデオ又は画像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化を含む）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を介したビデオ伝送、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。
【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一般制約情報のシグナリングと初期化に関

する。一例では、ビデオ復号化方法は、ビデオのビットストリームから一般制約情報（ＧＣＩ）フラグを抽出することと、ＧＣＩフラグの値に基づいて、ビデオに１つ又は複数の一般制約が課されていると決定することと、ビデオに１つ又は複数の一般制約が課されていると決定したことに応答して、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示す値を抽出することであって、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対応する追加符号化ツールを示す、ことと、当該値が５より大きいと決定することと、当該値が５より大きいと決定したことに応答して、ビットストリームから、６つの対応するフラグを表す６つのフラグビットを抽出することであって、これら６つの対応するフラグは、６つの追加符号化ツールの対応する制約を示すことと、これら６つのフラグによって示される６つの追加符号化ツールの制約に少なくとも部分的に基づいて、ビデオのビットストリームの残りの部分を画像に復号化することと、を含む。

【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一般制約情報（ＧＣＩ）フラグを抽出することと、ＧＣＩフラグの値に基づいて、ビデオに１つ又は複数の一般制約が課されていると決定することと、ビデオに１つ又は複数の一般制約が課されていると決定したことに応答して、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示す値を抽出することであって、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対応する追加符号化ツールを示す、ことと、当該値が５より大きいと決定することと、当該値が５より大きいと決定したことに応答して、ビットストリームから、６つの対応するフラグを表す６つのフラグビットを抽出することであって、これら６つの対応するフラグは、６つの追加符号化ツールの対応する制約を示すことと、これら６つのフラグによって示される６つの追加符号化ツールの制約に少なくとも部分的に基づいて、ビデオのビットストリームの残りの部分を画像に復号化することと、を含む。

【０００６】

さらに別の例では、システムは、処理機器と、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。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一般制約情報（ＧＣＩ）フラグを抽出することと、ＧＣＩフラグの値に基づいて、ビデオに１つ又は複数の一般制約が課されていると決定することと、ビデオに１つ又は複数の一般制約が課されていると決定したことに応答して、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示す値を抽出することであって、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対応する追加符号化ツールを示す、ことと、当該値が５より大きいと決定することと、当該値が５より大きいと決定したことに応答して、ビットストリームから、６つの対応するフラグを表す６つのフラグビットを抽出することであって、これら６つの対応するフラグは、６つの追加符号化ツールの対応する制約を示すことと、これら６つのフラグによって示される６つの追加符号化ツールの制約に少なくとも部分的に基づいて、ビデオのビットストリームの残りの部分を画像に復号化することと、を含む。

【０００７】

これらの例示的な実施例は、本発明を限定又は定義するのではなく、本発明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】本明細書で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】本明細書で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】本発明のいくつかの実施例に係る、ビデオ内の画像の符号化ツリーユニット分割の一例を示す。
【図４】本発明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一例を示す。
【図５】本発明のいくつかの実施例に係る、ビデオ復号化プロセスの一例を示す。
【図６】本発明のいくつかの実施例に係る、ビデオ復号化プロセスの別の例を示す。
【図７】本発明のいくつかの実施例に係る、ビデオ復号化プロセスの別の例を示す。
【図８】本発明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、本発明の特徴、実施例、及び利点をよりよく理解することができる。
【００１０】
ビデオ符号化のための一般制約情報のシグナリングと初期化の様々な実施例を提供する。上述のように、ますます多くのビデオデータが生成、記憶、及び伝送されている。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化できるように、ビデオ符号化の安定性を高めることも有益である。ビデオ復号化の安定性に関する問題には、非互換性と不整合の問題が含まれる。ビデオ符号化技術の発展に伴い、更新されたビデオ符号化規格が開発されている。このようなビデオ符号化規格の１つは、多用途ビデオ符号化（Ｖｅｒｓａｔｉｌｅ　Ｖｉｄｅｏ　Ｃｏｄｉｎｇ）規格のバージョン１であり、当該バージョン１は、国際標準化機構（ＩＳＯ）と国際電気通信連合（ＩＴＵ）によって共同で発行され、ここで、ＩＳＯによっては、「ＩＳＯ／ＩＥＣ　２３０９０－３：２０２１情報技術－イマ―シブメディアの符号化表示－パート３：多用途ビデオ符号化」として発行され、ＩＴＵによっては、「ＩＴＵ－Ｔ　Ｈ．２６６勧告（２０２０年８月）：多用途ビデオ符号化」として発行される。本発明では、多用途ビデオ符号化規格のバージョン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規格のバージョン２に置き換えられ、当該バージョン２は、ＩＳＯとＩＴＵによって共同で発行される予定であり、ここで、ＩＳＯによっては、「ＩＳＯ／ＩＥＣ　２３０９０－３：２０２２情報技術－イマ―シブメディアの符号化表示－パート３：多用途ビデオ符号化」として発行され、ＩＴＵによっては、「ＩＴＵ－Ｔ　Ｈ．２６６勧告（２０２２年４月）：多用途ビデオ符号化」として発行される。本発明では、多用途ビデオ符号化規格のバージョン２は、「ＶＶＣバージョン２」又は「ＶＶＣｖ２」とも呼ばれる。新しいバージョンのビデオ符号化規格に従うビデオデコーダが、以前のバージョンのビデオ符号化規格を使用して符号化されたビデオ信号を正常に復号化できるようにするためには、ビデオ符号化スキームを、以前のバージョンの符号化規格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一般制約情報のシグナリングは、ビデオ復号化で非同期化を引き起こし、これは、異なるバージョンのビデオ符号化規格間での重大な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、一般制約情報に関連する一般制約フラグが定義されていない可能性があり、デコーダ実装で曖昧さ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一般制約情報のシグナリングと初期化方法を導入してこれらの問題に対処することによって、ビデオ符号化の安定性を高める。
【００１１】
ＶＶＣ規格では、一般制約情報（ＧＣＩ：Ｇｅｎｅｒａｌ　Ｃｏｎｓｔｒａｉｎｔｓ　Ｉｎｆｏｒｍａｔｉｏｎ）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指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ビットストリームが一般制約情報をシグナリングし（即ち、ｇｃｉ＿ｐｒｅｓｅｎｔ＿ｆｌａｇの値が１である）、且つＶＶＣバージョン２一般制約情報が、制約される可能性のあるＮ個の追加符号化ツールで構成される場合、Ｎ個の追加符号化ツールに対応する構文要素ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓは、０又はＮの値にのみ設定できる。ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓが０に設定されている場合、Ｎ個の追加符号化ツールのための一般制約フラグはシグナリングされない。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次のＮビットは、Ｎ個の追加符号化ツールのための一般制約フラグをシグナリング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。

【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一般制約情報を処理できる。例えば、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓは、０より大きくＮより小さいＭ値、又はＮより大きい値に設定できる。Ｍが０より大きいがＮより小さい場合、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓ構文要素を復号化した後、デコーダは、ビットストリームからＭビットを抽出し、このＭビットを破棄する。ＭがＮより大きい場合、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓ構文要素を復号化した後、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ための一般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、これらの（Ｍ－Ｎ）ビットを破棄する。他の例では、ＶＶＣバージョン２デコーダは、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓを０より大きいがＮより小さい値に設定する一般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を０又はＮの値にのみ設定できる。将来のバージョンの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を許可しない。

【００１３】

いくつかの実施例において、一般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装の曖昧さと不整合の問題に対処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、且つｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓが０に等しい場合、ｇｅｎｅｒａｌ＿ｃｏｎｓｔｒａｉｎｔｓ＿ｉｎｆｏ（）は、一般制約情報フラグに関連する符号化ツールに制約を課さない。フラグの値が０で制約がないことを示す例では、一般制約情報フラグが存在しない場合、当該フラグの値が０に等しいと推測する。

【００１４】

本発明で説明される実施例は、以下の方法を提供し、当該方法によって、ＶＶＣバージョン２の追加符号化ツールのための一般制約フラグをシグナリング、推測することができる。先行技術とは異なり、本発明で説明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デコーダと互換性があり、ＶＶＣバージョン１デコーダとＶＶＣバージョン２デコーダの動作間の非同期化なしに復号化できる。本発明で説明される推測ルールは、ＶＶＣバージョン２ＧＣＩ構文要素のＶＶＣバージョン２復号化動作の曖昧さを除去する。これらの技術は、様々なビデオ符号化規格において効果的な符号化ツールとなり得る。

【００１５】

ここで図面を参照すると、図１は、本明細書で提供される実施例を実現するように構成されたビデオエンコーダ１００の例を示すブロック図である。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、インループ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、復号化画像バッファ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への入力は、一連のピクチャ（フレーム又は画像とも呼ばれる）を含む入力ビデオ１０２である。ブロックベースのビデオエンコーダでは、各画像について、ビデオエンコーダ１００は、分割モジュール１１２を用いて画像をブロック１０４に分割し、各ブロックは、複数の画素を含む。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枚の画像は、異なるサイズのブロックを含んでもよく、且つビデオの異なる画像のブロック分割も異なってもよい。各ブロックは、異なる予測（例えば、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）を使用して符号化されてもよい。

【００１７】

通常、ビデオ信号の１番目の画像は、イントラ符号化画像であり、当該画像は、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、同一画像からの符号化済みのデータのみを使用して画像のブロックを予測する。イントラ符号化の画像は、他の画像からの情報なしで復号化することができる。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を採用することができる。イントラ予測モジュール１２６は、同一画像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適用することによって、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、変換係数は、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によって変換されずに直接量子化されることができる。これは、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、その後に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を含む。このような量子化プロセスは、スカラー量子化と呼ばれる。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、独立して行うことができ、この量子化方法は、いくつかの既存のビデオ圧縮標準（例えば、Ｈ．２６４やＨＥＶＣなど）で使用される。Ｎ×Ｍブロックの場合、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。ブロック内の係数の量子化は、スキャン順序情報を使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、スキャン順序に沿った前の量子化値の状態に依存することができる。符号化効率をさらに高めるために、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／復号化スキャン順序において現在の係数に先行する情報に依存する。このような量子化方法は、依存量子化と呼ばれる。

【００２１】
量子化ステップサイズを使用して量子化の程度を調節することができる。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適用して、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、量子化は細かくなり、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）で示されることができる。ビデオの符号化ビットストリームで量子化パラメータが提供されるため、ビデオデコーダは、復号化のために量子化パラメータを取得して適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を用いて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、バイナリのｂｉｎに二値化され、符号化アルゴリズムによってこれらのバイナリのｂｉｎをさらにビットに圧縮する。二値化方法の例は、トランケーテッドライス（ＴＲ：ｔｒｕｎｃａｔｅｄ　Ｒｉｃｅ）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（ＥＧｋ：ｋ－ｔｈ　ｏｒｄｅｒ　Ｅｘｐ－Ｇｏｌｏｍｂ）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）、確率区間分割エントロピ（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】
上で説明したように、隣接ブロックからの再構成ブロック１３６は、画像のブロックのイントラ予測に使用される。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。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、再構成残差を決定することができる。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の場合、逆変換モジュール１１９はこれらのブロックに適用されない。逆量子化サンプルは、ブロックの再構成残差である。
【００２４】
インター予測又はイントラ予測を使用して、最初のイントラ予測画像に続く後続の画像のブロックを符号化することができる。インター予測では、画像内のブロックの予測は、１つ又は複数の以前に符号化されたビデオ画像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、動き推定のために、現在画像の現在ブロック１０４を復号化された参照画像１０８と比較する。復号化された参照画像１０８は、復号化画像バッファ１３０に記憶される。動き推定モジュール１２２は、復号化された参照画像１０８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１２２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１２４に提供される。場合によっては、複数の復号化された参照画像１０８において現在ブロックの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、動きベクトルは、対応する参照ブロックと共にインター予測モジュール１２４に提供される。

【００２６】
インター予測モジュール１２４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１３４を生成する。例えば、インター予測モジュール１２４は、動きベクトルに基づいて、対応する参照画像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いくつかの重みと組み合わせて、現在ブロックの予測ブロック１３４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１００は、ブロック１０４からインター予測ブロック１３４を減算して残差ブロック１０６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１０６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、インター予測ブロックの再構成ブロック１３６は、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１３４と組み合わせることによって得られる。

【００２８】
動き推定に使用される復号化された画像１０８を得るために、再構成ブロック１３６をインループフィルタモジュール１２０によって処理する。インループフィルタモジュール１２０は、画素遷移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。インループフィルタモジュール１２０は、ディブロッキングフィルタ、サンプル適応オフセット（ＳＡＯ）フィルタ、適応ループフィルタ（ＡＬＦ）などの、１つ又は複数のインループフィルタを実装するように構成される。

【００２９】
図２は、本明細書で提供される実施例を実現するように構成されたビデオデコーダ２００の一例を示す。ビデオデコーダ２００は、ビットストリームにおける符号化済みのビデオ２０２を処理し、復号化された画像２０８を生成する。図２に示す例では、ビデオデコーダ２００は、エントロピ復号化モジュール２１６、逆量子化モジュール２１８、逆変換モジュール２１９、インループフィルタモジュール２２０、イントラ予測モジュール２２６、インター予測モジュール２２４、及び復号化画像バッファ２３０を備える。

【００３０】
エントロピ復号化モジュール２１６は、符号化済みのビデオ２０２のエントロピ復号化を実行するように構成される。エントロピ復号化モジュール２１６は、量子化係数、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符号化パラメータ、及び他の情報を復号化する。いくつかの例では、エントロピ復号化モジュール２１６は、符号化済みのビデオ２０２のビットストリームをバイナリ表現に復号化し、その後、バイナリ表現を係数の量子化レベルに変換する。その後、エントロピ復号化された係数レベルは、逆量子化モジュール２１８によって逆量子化され、続いて、逆変換モジュール２１９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２１８と逆変換モジュール２１９の機能はそれぞれ、上記の図１で説明された逆量子化モジュール１１８と逆変換モジュール１１９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２３６を生成するために

、対応する予測ブロック２３４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の場合、逆変換モジュール２１９はこれらのブロックに適用されない。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、ブロックの予測モードに基づいて生成される。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、同じ画像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。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。

【００３２】

図１に関して上記で説明したように、インター予測には、１つ又は複数の参照画像が関係する。ビデオデコーダ２００は、インループフィルタモジュール２２０を参照画像の再構成ブロックに適用することにより、参照画像の復号化された画像２０８を生成する。復号化された画像２０８は、インター予測モジュール２２４によって使用されるために、また出力のために、復号化画像バッファ２３０に記憶される。

【００３３】

次に図３を参照すると、図３は、本発明のいくつかの実施例に係る、ビデオ内の画像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に関して上記で説明したように、ビデオの画像を符号化するために、画像は、図３に示すように、ＶＶＣにおけるＣＴＵ（符号化ツリーユニット）３０２のようなブロックに分割される。例えば、ＣＴＵ３０２は、１２８×１２８画素のブロックであってもよい。ＣＴＵは、何らかの順序（例えば、図３に示す順序）に従って処理される。いくつかの例では、画像内の各ＣＴＵ３０２は、図４に示す１つ又は複数のＣＵ（符号化ユニット）４０２に分割されることができ、ＣＵ４０２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、ＣＴＵ３０２は、異なる方式でＣＵ４０２に分割されることができる。例えば、ＶＶＣでは、ＣＵ４０２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することができる。各ＣＵ４０２は、そのルートＣＴＵ３０２と同じサイズであってもよいし、ルートＣＴＵ３０２を４×４ブロックほど小さいブロックに分割したものになってもよい。図４に示すように、ＶＶＣでＣＴＵ３０２をＣＵ４０２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。

【００３４】

ＶＶＣバージョン１（ＶＶＣｖ１）における一般制約情報について

ＶＶＣバージョン１では、一般制約情報（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指定するために使用される。ｇｃｉ＿ｐｒｅｓｅｎｔ＿ｆｌａｇｆ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指定する。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（）構文構造がいかなる制約も課さないことを指定する。

【００３５】

複数のコンテキストにおいて、高レベル構文で一般制約情報をシグナリングすることが

できる。例えば、復号化能力情報のみを含むネットワークパケットでＧＣＩをシグナリングすることができ、このようなネットワークパケットは、例えば、復号化能力情報のみを搬送し、そのｎａｌ＿ｕｎｉｔ＿ｔｙｐｅが１３に設定された（即ち、ｎａｌ＿ｕｎｉｔ＿ｔｙｐｅの名称としてＤＣＩ＿ＮＵＴに設定された）ＮＡＬ（Ｎｅｔｗｏｒｋ　Ａｂｓｔｒａｃｔｉｏｎ　Ｌａｙｅｒ）パケットである。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をシグナリングすることができる。

【００３６】

ＧＣＩ構文構造の目的は、ビットストリームの復号化に必要な特徴に関する構成情報の検出を可能にし、相互運用点（ＩＯＰ：ｉｎｔｅｒｏｐｅｒａｂｉｌｉｔｙ　ｐｏｉｎｔ）をシグナリングできるようにすることであり、前記相互運用点は、ＰＴＬ（Ｐｒｏｆｉｌｅ，Ｔｉｅｒ　ａｎｄ　Ｌｅｖｅｌ）で指定された限制以外の限制を課し、その粒度は、以前のビデオ符号化規格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、ＶＶＣプロファイルのすべての特徴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に対応するデコーダ実現の相互運用性を定義することを可能にすることができる。デコーダ実現では、ＧＣＩ構文要素を調べて、ビットストリームが特定の特徴の使用を回避しているか否かをチェックし、復号化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化可能か否かを確認することができる。ＶＶＣプロファイルのすべての特徴をサポートするデコーダ実現では、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に準拠するすべてのビットストリームを復号化できるためである。

【００３７】

ＶＶＣバージョン１で指定された一般制約情報構文構造は、次のように定義される。

【００３８】

【表１（１）】

general_constraints_info( ) {	**記述子**
gci_present_flag	u(1)
if(gci_present_flag) {	
/* general */	
gci_intra_only_constraint_flag	u(1)
gci_all_layers_independent_constraint_flag	u(1)
gci_one_au_only_constraint_flag	u(1)
/* picture format */	
gci_sixteen_minus_max_bitdepth_constraint_idc	u(4)
gci_three_minus_max_chroma_format_constraint_idc	u(2)
/* 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)
/* 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)

【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 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)
/* 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)
/* 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)

【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)
/* 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_cbcr_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)
/* 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)

【表１（４）】

gci_no_virtual_boundaries_constraint_flag	u(1)
gci_num_reserved_bits	u(8)
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)
}	

【００３９】
一般制約フラグの存在は、ｇｃｉ＿ｐｒｅｓｅｎｔ＿ｆｌａｇの値に依存する。ｇｃｉ＿ｐｒｅｓｅｎｔ＿ｆｌａｇの値が１である場合、ビットストリームに一般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一般制約フラグが存在しない。
【００４０】
ＶＶＣｖ１で定義された一般制約フラグ以外に、構文要素ｇｃｉ＿ｎｕｍ＿ｒｅｓｅｒｖｅｄ＿ｂｉｔｓによって追加の一般制約フラグを提供することができる。ｇｃｉ＿ｎｕｍ＿ｒｅｓｅｒｖｅｄ＿ｂｉｔｓは、８ビットの符号なし整数で、その値は、一般制約構文構造でシグナリング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。
【００４１】
ＶＶＣバージョン２（ＶＶＣｖ２）における一般制約情報について
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、ＪＶＥＴ－Ｘ２００５の共同ビデオエキスパートチームのドキュメント）では、いくつかの追加の符号化ツールが一般制約フラグによって制約されることが提案されている。ＶＶＣｖ１でｇｃ＿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。提案されているＶＶＣｖ２の調整された構文は、次の通りである。
【００４２】

【表２】

general_constraints_info( ) {	記述子
gci_present_flag	u(1)
if(gci_present_flag) {	
/* 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)
}	

次の合計６つの追加の一般制約フラグがあり、それぞれ、ｇｃｉ＿ａｌｌ＿ｒａｐ＿ｐｉｃｔｕｒｅｓ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇ、ｇｃｉ＿ｎｏ＿ｅｘｔｅｎｄｅｄ＿ｐｒｅｃｉｓｉｏｎ＿ｐｒｏｃｅｓｓｉｎｇ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇ、ｇｃｉ＿ｎｏ＿ｔｓ＿ｒｅｓｉｄｕａｌ＿ｃｏｄｉｎｇ＿ｒｉｃｅ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇ、ｇｃｉ＿ｎｏ＿ｒｒｓ＿ｒｉｃｅ＿ｅｘｔｅｎｓｉｏｎ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇ、ｇｃｉ＿ｎｏ＿ｐｅｒｓｉｓｔｅｎｔ＿ｒｉｃｅ＿ａｄａｐｔａｔｉｏｎ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇ、及びｇｃｉ＿ｎｏ＿ｒｅｖｅｒｓｅ＿ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇである。提案されたｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓ構文要素の解釈（「セマンティック」）は、次の通りで

ある。
【００４３】
ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓは、ｇｃｉ＿ａｌｉｇｎｍｅｎｔ＿ｚｅｒｏ＿ｂｉｔ構文要素（存在する場合）以外の一般制約情報構文構造内の追加ＧＣＩビットの数を指定する。当該ドキュメントのこの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。ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの１より大きい値は、ＩＴＵ－Ｔ｜ＩＳＯ／ＩＥＣによる将来の使用のために予約されている。当該ドキュメントのこのバージョンでは、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値が０又は１に等しくなるように要求されているが、当該ドキュメントのこのバージョンに準拠するデコーダは、構文に１より大きいｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値が表示されることを許可し、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。
【００４４】
ＶＶＣｖ２ビットストリームが一般制約情報をシグナリングする場合、６つのＶＶＣｖ２一般制約フラグがシグナリングされていない場合は、構文要素ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓを０に設定し、６つのＶＶＣｖ２一般制約フラグがシグナリングされている場合は、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓを１に設定することが提案された。
【００４５】
しかし、上記の提案された一般制約構文のＶＶＣｖ２仕様は、ＶＶＣｖ１デコーダとの非互換性を引き起こす。具体的には、追加の一般制約フラグがシグナリングされると、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一般制約フラグをシグナリングする。ＶＶＣｖ２デコーダでは、一般制約情報がシグナリングされると、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓ構文要素は、ビットストリームから８ビットの符号なし整数に復号化される。ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓが値１に復号化されると、ビットストリームから６つの追加ビットが復号化される。これらの６ビットは、ＶＶＣｖ２で制約される可能性のある追加の復号化ツールの一般制約フラグとして解釈される。
【００４６】
ＶＶＣｖ１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。しかし、当該構文要素が値１に復号化されると、ビットストリームから１つの追加ビットのみが復号化される。当該ビットは破棄され、使用されない。その結果、ＶＶＣｖ２ビットストリームを復号化するＶＶＣｖ１デコーダは、ＶＶＣｖ１仕様で認識されない５つの追加ビットがビットストリームに存在するため、非同期化の問題に遭遇する可能性がある。
【００４７】
上記で説明したように、非同期化は、異なるバージョンのビデオ符号化規格間での重大な非互換性の問題である。ＶＶＣｖ１デコーダは、ＶＶＣｖ２ビットストリーム全体を復号化できない可能性があり、これは、ＶＶＣｖ２ビットストリームが、ＶＶＣｖ２仕様で定義されているが、ＶＶＣｖ１デコーダには知られていない符号化ツールを使用する可能性があるためである。しかし、ＶＶＣｖ１デコーダは、少なくともＶＶＣｖ２ビットストリームの高レベル構文部分を復号化できることが望ましい。高レベル構文を正常に復号化することにより、ビデオデコーダは、一般制約情報だけでなく、プロファイルと階層情報（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化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、一般制約情報は、どの符号化ツールがビットストリームによって制約されるかに関する指示をデコーダに提供する。プロファイルと階層情報は、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。
【００４８】
高レベル構文が正常に復号化された場合、ビデオデコーダは、現在のビットストリームを復号化できるか否かを決定することができ、復号化できない場合、デコーダは、復号化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化中に非同期化が発生すると、高レベル構文で提供される情報が正しく復号化されない可能性がある。最悪の場合、デコーダは、非同期化イベントに続く構文要素の完全に誤った値を復号化する可能性があり、その結果、誤ったパラメータ設定や、後続の低レベル構文の誤った復号化につながり、復号化が失敗する可能性がある。
【００４９】
さらに、ＶＶＣｖ２デコーダでは、一般制約情報がシグナリングされると、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓ構文要素が、ビットストリームから８ビットの符号なし整数に復号化される。ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓが値０に復号化された場合、それ以上の一般制約フラグはシグナリングされない。この場合、追加の一般制約フラグに対して推測値が指定されず、追加の一般制約フラグの値は未定義である。したがって、追加の一般制約フラグに関連する符号化ツールを制約するか制約しないかの動作は曖昧であり、デコーダの実現に一貫性がなくなる可能性がある。
【００５０】
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、予約された構文要素がエンコーダによってビットストリームにプレースホルダ（ｐｌａｃｅｈｏｌｄｅｒ）として書き込まれる場合、デフォルト値は、予約された構文要素の名前に埋め込まれる。しかし、一般制約構文構造の設計により、ｇｃｉ＿ｒｅｓｅｒｖｅｄ＿ｚｅｒｏ＿ｂｉｔ［ｉ］はエンコーダによって書き込まれることはない。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。
【００５１】
一般制約情報のシグナリングについて
上記の非同期化の問題を解決するために一般制約情報をシグナリングする１つの実施例では、ＶＶＣｖ２ビットストリームが一般制約情報をシグナリングし（即ち、ｇｃｉ＿ｐｒｅｓｅｎｔ＿ｆｌａｇの値が１である）、且つＶＶＣｖ２一般制約情報が制約される可能性のあるＮ個の追加符号化ツールで構成される場合、構文要素ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓは、０又はＮの値にのみ設定できる。ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓが０に設定された場合、Ｎ個の追加符号化ツールの一般制約フラグはシグナリングされない。ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓがＮに設定された場合、ビットストリーム内の次のＮビットは、Ｎ個の追加符号化ツールの一般制約フラグをシグナリング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。
【００５２】
ＶＶＣｖ２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デコーダは、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓを０又はＮ以外の値に設定する一般制約情報を処理できる。例えば、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓは、０より大きいが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の復号化された値をＭとする。そして、Ｍが０より大きいがＮより小さい（即ち、０＜Ｍ＜Ｎ）場合、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓ構文要素を復号化した後、デコーダは、ビットストリームからｇｃｉ＿ｒｅｓｅｒｖｅｄ＿ｚｅｒｏ＿ｂｉｔ［ｉ］構文要素としてＭビットを抽出し、このＭビットを破棄する。ＭがＮより大きい（Ｎ＜Ｍ）場合、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓ

構文要素を復号化した後、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ための一般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ｇｃｉ＿ｒｅｓｅｒｖｅｄ＿ｚｅｒｏ＿ｂｉｔ［ｉ］構文要素として（Ｍ－Ｎ）ビットをさらに抽出し、これらの（Ｍ－Ｎ）ビットを破棄する。

【００５３】

他の例では、ＶＶＣｖ２デコーダは、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓを０より大きいがＮより小さい値に設定する一般制約情報を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を０の値にのみ設定できる。正当なＶＶＣｖ２ビットストリームは、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓを０又はＮの値にのみ設定できる。将来のバージョンの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。

【００５４】

当該実施例の１つの例では、ＶＶＣｖ２符号化ツールのために現在提案されている６つの一般制約フラグを使用したＶＶＣｖ２の一般制約情報構文の変更は、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））、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。

【００５５】

【表３】

表１

general_constraints_info( ) {	**記述子**
gci_present_flag	u(1)
if(gci_present_flag) {	
/*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)
}	

【００５６】

１つの例では、符号化ツールのための６つの追加の一般制約フラグがＶＶＣｖ２に存在する場合、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、削除された内容は取り消し線付きで示される）。

【００５７】

ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓは、ｇｃｉ＿ａｌｉｇｎｍｅｎｔ＿ｚｅｒｏ＿ｂｉｔ構文要素（存在する場合）以外の一般制約情報構文構造内の追加ＧＣＩビットの数を指定する。当該ドキュメントのこの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又は６以外の値は、ＩＴＵ－Ｔ｜ＩＳＯ／ＩＥＣによる将来の使用のために予約されている。当該ドキュメントのこのバージョンでは、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値は０又は≪１≫６に等しくなるように要求されているが、当該ドキュメントのこのバージョンに準拠するデコーダは、構文に≪１より大きい≫０又は６以外のｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値が表示されることを許可し、且つｇｃｉ＿ｎｕｍ

＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓが≪１より大きい≫０又は６以外である場合は、すべてのｇｃｉ＿ｒｅｓｅｒｖｅｄ＿ｚｅｒｏ＿ｂｉｔ［ｉ］構文要素の値を無視すべきである。

【００５８】

ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓのセマンティックの上記の例では、上記で説明した０又は６の値に加えて、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓは、０又は６以外の値を取ることが許可される。つまり、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓは、１～５の値を取ることができる。ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓが６より大きい値を取ることも許可される。ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓがＭで示される１～５の値を有する場合、上記の表１に示す構文に従って、デコーダは、「ｉｆ」条件がｔｕｒｅであるときに実行されるステップをスキップし、「ｅｌｓｅ」ステップにジャンプして、「ｎｕｍＡｄｄｉｔｉｏｎａｌＢｉｔｓＵｓｅｄ」の値を０に割り当てる。次に、Ｍビットが「ｆｏｒ」ループでＭビットを読み取られ、破棄される。このようにして、非同期化の問題を回避することができる。ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値Ｍが６より大きい場合、６つの追加の一般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。

【００５９】

別の例では、符号化ツールのための６つの追加の一般制約フラグがＶＶＣｖ２に存在する場合、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、削除された内容は取り消し線付きで示される）。

【００６０】

ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓは、ｇｃｉ＿ａｌｉｇｎｍｅｎｔ＿ｚｅｒｏ＿ｂｉｔ構文要素（存在する場合）以外の一般制約情報構文構造内の追加ＧＣＩビットの数を指定する。当該ドキュメントのこの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≫６より大きい値は、ＩＴＵ－Ｔ｜ＩＳＯ／ＩＥＣによる将来の使用のために予約されている。当該ドキュメントのこのバージョンでは、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値が０又は≪１≫６に等しくなるように要求されているが、当該ドキュメントのこのバージョンに準拠するデコーダは、構文に≪１≫６より大きいｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値が表示されることを許可し、且つｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓが≪１≫６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。

【００６１】

一般制約情報をシグナリングする別の実施例では、ＶＶＣｖ２ビットストリームが一般制約情報をシグナリングし（即ち、ｇｃｉ＿ｐｒｅｓｅｎｔ＿ｆｌａｇの値が１である）、且つＶＶＣｖ２一般制約情報が制約される可能性のあるＮ個の追加符号化ツールで構成される場合、構文要素ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓは、Ｍの値に設定できる。Ｍは、０からＮ（端点を含む）の範囲（０≦Ｍ≦Ｎ）である。ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓが０に設定された場合、Ｎ個の追加符号化ツールのための一般制約フラグはシグナリングされない。ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓが非ゼロ値Ｍに設定された場合、ビットストリームの次のＭビットは、Ｎ個の追加符号化ツールのうちのＭ個の追加符号化ツールの一般制約フラグをシグナリング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、一般制約フラグが一般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。

【００６２】

当該実施例の１つの例では、ＶＶＣｖ２符号化ツールのために現在提案されている６つの一般制約フラグを使用したＶＶＣｖ２の一般制約情報構文の変更は、次の通りであり得る。

【００６３】
【表４】

general_constraints_info( ) {	記述子
gci_present_flag	u(1)
if(gci_present_flag) {	
/* 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)
}	

【００６４】
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を使用して同等の動作を実現することができる。
【００６５】

【表５】

general_constraints_info( ) {	記述子
gci_present_flag	u(1)
if(gci_present_flag) {	
/* 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)
}	

【００６６】

ＶＶＣｖ２符号化ツールのための一般制約フラグの順序を変更することによって、一般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。

【００６７】

符号化ツールのための６つの追加の一般制約フラグがＶＶＣｖ２に存在する場合、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの対応するセマンティックは、次の通りであり得る。

【００６８】

ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓは、ｇｃｉ＿ａｌｉｇｎｍｅｎｔ＿ｚｅｒｏ＿ｂｉｔ構文要素（存在する場合）以外の一般制約情報構文構造内の追加ＧＣＩビットの数を指定する。当該ドキュメントのこの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（端点を含む）内にあるべきである。ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの６より大

きい値は、ＩＴＵ－Ｔ｜ＩＳＯ／ＩＥＣによる将来の使用のために予約されている。当該ドキュメントのこのバージョンでは、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値は０～６の範囲（端点を含む）内にあるように要求されているが、当該ドキュメントのこのバージョンに準拠するデコーダは、構文に６より大きいｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値が表示されることを許可し、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。

【００６９】

当該実施例では、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値に応じて、追加の一般制約フラグの一部又はすべてがシグナリングされない場合がある。当該実施例の１つの配置では、ＶＶＣｖ２ビットストリームが一般制約情報をシグナリングする場合（即ち、ｇｃｉ＿ｐｒｅｓｅｎｔ＿ｆｌａｇの値が１である場合）、シグナリングされない追加の一般制約フラグに対応する符号化ツールには制約が課されない。この動作は、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓのセマンティックを次のように変更することによって表現できる。

【００７０】

ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓは、ｇｃｉ＿ａｌｉｇｎｍｅｎｔ＿ｚｅｒｏ＿ｂｉｔ構文要素（存在する場合）以外の一般制約情報構文構造内の追加ＧＣＩビットの数を指定する。当該ドキュメントのこの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（端点を含む）内にあるべきである。ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの６より大きい値は、ＩＴＵ－Ｔ｜ＩＳＯ／ＩＥＣによる将来の使用のために予約されている。当該ドキュメントのこのバージョンでは、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値が０～６の範囲（端点を含む）内にあるように要求されているが、当該ドキュメントのこのバージョンに準拠するデコーダは、構文に６より大きいｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値が表示されることを許可し、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が課されない。

【００７１】

当該実施例の別の１つの配置では、ＶＶＣｖ２ビットストリームが一般制約情報をシグナリングし（即ち、ｇｃｉ＿ｐｒｅｓｅｎｔ＿ｆｌａｇの値が１である場合）、且つ追加の一般制約フラグがシグナリングされない場合、対応するツールが制約される。この動作は、追加の一般制約フラグのセマンティックを次のように変更することによって表現できる。

【００７２】

ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓは、ｇｃｉ＿ａｌｉｇｎｍｅｎｔ＿ｚｅｒｏ＿ｂｉｔ構文要素（存在する場合）以外の一般制約情報構文構造内の追加ＧＣＩビットの数を指定する。当該ドキュメントのこの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（端点を含む）内にあるべきである。ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの６より大きい値は、ＩＴＵ－Ｔ｜ＩＳＯ／ＩＥＣによる将来の使用のために予約されている。当該ドキュメントのこのバージョンでは、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値が０～６の範囲（端点を含む）内にあるように要求されているが、当該ドキュメントのこのバージョンに準拠するデコーダは、構文に６より大きいｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値が表示されることを許可し、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。

【００７３】

gci_all_rap_pictures_constraint_flagが１に等しい場合、OlsInScope内のすべての画像がＩＲＡＰ画像、又はph_recovery_poc_cntが０に等しいＧＤＲ画像であることを指定する。gci_all_rap_pictures_constraint_flagが０に等しい場合、このような制約は課されない。gci_present_flagが１に等しく、gci_all_rap_pictures_constraint_flagが存在しない場合、gci all rap pictures constraint flagの値が１に等しいと推測する。

【００７４】

gci_no_extended_precision_processing_constraint_flagが１に等しい場合、OlsInScope内のすべての画像のsps_extended_precision_flagが０に等しくなることを指定する。gci_no_extended_precision_processing_constraint_flagが０に等しい場合、このような制約は課されない。gci_present_flagが１に等しく、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に等しいと推測する。

【００７５】

gci_no_ts_residual_coding_rice_constraint_flagが１に等しい場合、OlsInScope内のすべての画像のsps_ts_residual_coding_rice_present_in_sh_flagが０に等しくなることを指定する。gci_no_ts_residual_coding_rice_constraint_flagが０に等しい場合、このような制約は課されない。gci_present_flagが１に等しく、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に等しいと推測する。

【００７６】

gci_no_rrc_rice_extension_constraint_flagが１に等しい場合、OlsInScope内のすべての画像のsps_rrc_rice_extension_flagが０に等しくなることを指定する。gci_no_rrc_rice_extension_constraint_flagが０に等しい場合、このような制約は課されない。gci_present_flagが１に等しく、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に等しいと推測する。

【００７７】

gci_no_persistent_rice_adaptation_constraint_flagが１に等しい場合、OlsInScope内のすべての画像のsps_persistent_rice_adaptation_enabled_flagが０に等しくなることを指定する。gci_no_persistent_rice_adaptation_constraint_flagが０に等しい場合、このような制約は課されない。gci_present_flagが１に等しく、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に等しいと推測する。

【００７８】

gci_no_reverse_last_sig_coeff_constraint_flagが１に等しい場合、OlsInScope内のすべての画像のsps_re

ｖｅｒｓｅ＿ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｅｎａｂｌｅｄ＿ｆｌａｇが０に等しくなることを指定する。ｇｃｉ＿ｎｏ＿ｒｅｖｅｒｓｅ＿ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが０に等しい場合、このような制約は課されない。ｇｃｉ＿ｐｒｅｓｅｎｔ＿ｆｌａｇが１に等しく、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に等しいと推測する。

【００７９】

一般制約情報フラグの初期化について

上述のデコーダ実装の曖昧さと不整合の問題に対処するために、一般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、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課さない。

【００８０】

一例として、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓのセマンティックに対する可能な変更は、次に示す通りであり、これは、現在バージョンのＶＶＣバージョン２仕様に基づいて行われる（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）。

【００８１】

ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓは、ｇｃｉ＿ａｌｉｇｎｍｅｎｔ＿ｚｅｒｏ＿ｂｉｔ構文要素（存在する場合）以外の一般制約情報構文構造内の追加ＧＣＩビットの数を指定する。当該ドキュメントのこの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。ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの１より大きい値は、ＩＴＵ－Ｔ｜ＩＳＯ／ＩＥＣによる将来の使用のために予約されている。当該ドキュメントのこのバージョンでは、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値が０又は１に等しくなるように要求されているが、当該ドキュメントのこのバージョンに準拠するデコーダは、構文に１より大きいｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値が表示されることを許可し、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、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課さない。

【００８２】

別の例として、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓのセマンティックに

対する可能な変更は、次に示す通りであり、これは、現在バージョンのＶＶＣバージョン２仕様に基づいて行われる。

【００８３】

ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓは、ｇｃｉ＿ａｌｉｇｎｍｅｎｔ＿ｚｅｒｏ＿ｂｉｔ構文要素（存在する場合）以外の一般制約情報構文構造内の追加ＧＣＩビットの数を指定する。当該ドキュメントのこの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又は６以外の値は、ＩＴＵ－Ｔ｜ＩＳＯ／ＩＥＣによる将来の使用のために予約されている。当該ドキュメントのこのバージョンでは、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値が０又は≪１≫６に等しくなるように要求されているが、当該ドキュメントのこのバージョンに準拠するデコーダは、構文に≪１より大きい≫０又は６以外のｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値が表示されることを許可し、且つｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓが≪１より大きい≫０又は６以外である場合は、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、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課さない。

【００８４】

現在バージョンのＶＶＣバージョン２仕様では、ｇｃｉ＿ｐｒｅｓｅｎｔ＿ｆｌａｇが１に等しい場合にのみ、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓのセマンティックが使用される。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が１～５の値を取ることは許可されていないため、「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。

【００８５】

ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓは、ｇｃｉ＿ａｌｉｇｎｍｅｎｔ＿ｚｅｒｏ＿ｂｉｔ構文要素（存在する場合）以外の一般制約情報構文構造内の追加ＧＣＩビットの数を指定する。当該ドキュメントのこの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又は６

以外の値は、ＩＴＵ－Ｔ｜ＩＳＯ／ＩＥＣによる将来の使用のために予約されている。当該ドキュメントのこのバージョンでは、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値が０又は≪１≫６に等しくなるように要求されているが、当該ドキュメントのこのバージョンに準拠するデコーダは、構文に≪１より大きい≫０又は６以外のｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値が表示されることを許可し、且つｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓが≪１より大きい≫０又は６以外である場合は、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＿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課さない。

【００８６】

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一般制約フラグの推測値を設定できる。例えば、追加の一般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。

【００８７】

ｇｃｉ＿ａｌｌ＿ｒａｐ＿ｐｉｃｔｕｒｅｓ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが１に等しい場合、ＯｌｓＩｎＳｃｏｐｅ内のすべての画像がＩＲＡＰ画像、又はｐｈ＿ｒｅｃｏｖｅｒｙ＿ｐｏｃ＿ｃｎｔが０に等しいＧＤＲ画像であることを指定する。ｇｃｉ＿ａｌｌ＿ｒａｐ＿ｐｉｃｔｕｒｅｓ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが０に等しい場合、このような制約は課されない。存在しない場合、ｇｃｉ＿ａｌｌ＿ｒａｐ＿ｐｉｃｔｕｒｅｓ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇの値が０に等しいと推測する。

【００８８】

ｇｃｉ＿ｎｏ＿ｅｘｔｅｎｄｅｄ＿ｐｒｅｃｉｓｉｏｎ＿ｐｒｏｃｅｓｓｉｎｇ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが１に等しい場合、ＯｌｓＩｎＳｃｏｐｅ内のすべての画像のｓｐｓ＿ｅｘｔｅｎｄｅｄ＿ｐｒｅｃｉｓｉｏｎ＿ｆｌａｇが０に等しくなることを指定する。ｇｃｉ＿ｎｏ＿ｅｘｔｅｎｄｅｄ＿ｐｒｅｃｉｓｉｏｎ＿ｐｒｏｃｅｓｓｉｎｇ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが０に等しい場合、このような制約は課されない。存在しない場合、ｇｃｉ＿ｎｏ＿ｅｘｔｅｎｄｅｄ＿ｐｒｅｃｉｓｉｏｎ＿ｐｒｏｃｅｓｓｉｎｇ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇの値が０に等しいと推測する。

【００８９】

ｇｃｉ＿ｎｏ＿ｔｓ＿ｒｅｓｉｄｕａｌ＿ｃｏｄｉｎｇ＿ｒｉｃｅ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが１に等しい場合、ＯｌｓＩｎＳｃｏｐｅ内のすべての画像のｓｐｓ＿ｔｓ＿ｒｅｓｉｄｕａｌ＿ｃｏｄｉｎｇ＿ｒｉｃｅ＿ｐｒｅｓｅｎｔ＿ｉｎ＿ｓｈ＿ｆｌａｇが０に等しくなることを指定する。ｇｃｉ＿ｎｏ＿ｔｓ＿ｒｅｓｉｄｕａｌ＿ｃｏｄｉｎｇ＿ｒｉｃｅ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが０に等しい場合、このような制約は課されない。存在しない場合、ｇｃｉ＿ｎｏ＿ｔｓ＿ｒｅｓｉｄｕａｌ＿ｃｏｄｉｎｇ＿ｒｉｃｅ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇの値が０に等しいと推測する。

【００９０】
ｇｃｉ＿ｎｏ＿ｒｒｃ＿ｒｉｃｅ＿ｅｘｔｅｎｓｉｏｎ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが１に等しい場合、ＯｌｓＩｎＳｃｏｐｅ内のすべての画像のｓｐｓ＿ｒｒｃ＿ｒｉｃｅ＿ｅｘｔｅｎｓｉｏｎ＿ｆｌａｇが０に等しくなることを指定する。ｇｃｉ＿ｎｏ＿ｒｒｃ＿ｒｉｃｅ＿ｅｘｔｅｎｓｉｏｎ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが０に等しい場合、このような制約は課されない。存在しない場合、ｇｃｉ＿ｎｏ＿ｒｒｃ＿ｒｉｃｅ＿ｅｘｔｅｎｓｉｏｎ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇの値が０に等しいと推測する。
【００９１】
ｇｃｉ＿ｎｏ＿ｐｅｒｓｉｓｔｅｎｔ＿ｒｉｃｅ＿ａｄａｐｔａｔｉｏｎ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが１に等しい場合、ＯｌｓＩｎＳｃｏｐｅ内のすべての画像のｓｐｓ＿ｐｅｒｓｉｓｔｅｎｔ＿ｒｉｃｅ＿ａｄａｐｔａｔｉｏｎ＿ｅｎａｂｌｅｄ＿ｆｌａｇが０に等しくなることを指定する。ｇｃｉ＿ｎｏ＿ｐｅｒｓｉｓｔｅｎｔ＿ｒｉｃｅ＿ａｄａｐｔａｔｉｏｎ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが０に等しい場合、このような制約は課されない。存在しない場合、ｇｃｉ＿ｎｏ＿ｐｅｒｓｉｓｔｅｎｔ＿ｒｉｃｅ＿ａｄａｐｔａｔｉｏｎ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇの値が０に等しいと推測する。
【００９２】
ｇｃｉ＿ｎｏ＿ｒｅｖｅｒｓｅ＿ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが１に等しい場合、ＯｌｓＩｎＳｃｏｐｅ内のすべての画像のｓｐｓ＿ｒｅｖｅｒｓｅ＿ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｅｎａｂｌｅｄ＿ｆｌａｇが０に等しくなることを指定する。ｇｃｉ＿ｎｏ＿ｒｅｖｅｒｓｅ＿ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが０に等しい場合、このような制約は課されない。存在しない場合、ｇｃｉ＿ｎｏ＿ｒｅｖｅｒｓｅ＿ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇの値が０に等しいと推測する。
【００９３】
別の例では、追加の一般制約フラグのセマンティック（）は、次のように変更される。
【００９４】
ｇｃｉ＿ａｌｌ＿ｒａｐ＿ｐｉｃｔｕｒｅｓ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが１に等しい場合、ＯｌｓＩｎＳｃｏｐｅ内のすべての画像がＩＲＡＰ画像、又はｐｈ＿ｒｅｃｏｖｅｒｙ＿ｐｏｃ＿ｃｎｔが０に等しいＧＤＲ画像であることを指定する。ｇｃｉ＿ａｌｌ＿ｒａｐ＿ｐｉｃｔｕｒｅｓ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが０に等しい場合、このような制約は課されない。ｇｃｉ＿ａｌｌ＿ｒａｐ＿ｐｉｃｔｕｒｅｓ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが存在しない場合、ｇｃｉ＿ａｌｌ＿ｒａｐ＿ｐｉｃｔｕｒｅｓ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇの値が０に等しいと推測する。
【００９５】
ｇｃｉ＿ｎｏ＿ｅｘｔｅｎｄｅｄ＿ｐｒｅｃｉｓｉｏｎ＿ｐｒｏｃｅｓｓｉｎｇ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが１に等しい場合、ＯｌｓＩｎＳｃｏｐｅ内のすべての画像のｓｐｓ＿ｅｘｔｅｎｄｅｄ＿ｐｒｅｃｉｓｉｏｎ＿ｆｌａｇが０に等しくなることを指定する。ｇｃｉ＿ｎｏ＿ｅｘｔｅｎｄｅｄ＿ｐｒｅｃｉｓｉｏｎ＿ｐｒｏｃｅｓｓｉｎｇ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが０に等しい場合、このような制約は課されない。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０に等しいと推測する。
【００９６】
ｇｃｉ＿ｎｏ＿ｔｓ＿ｒｅｓｉｄｕａｌ＿ｃｏｄｉｎｇ＿ｒｉｃｅ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが１に等しい場合、ＯｌｓＩｎＳｃｏｐｅ内のすべての画像のｓｐｓ＿ｔｓ＿ｒｅｓｉｄｕａｌ＿ｃｏｄｉｎｇ＿ｒｉｃｅ＿ｐｒｅｓｅｎｔ＿ｉｎ＿ｓｈ＿ｆｌａｇが０に等しくなることを指定する。ｇｃｉ＿ｎｏ＿ｔｓ＿ｒｅｓｉｄｕａｌ＿ｃｏｄｉｎｇ＿ｒｉｃｅ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが０に等しい場合、このような制約は課されない。ｇｃｉ＿ｎｏ＿ｔｓ＿ｒｅｓｉｄｕａｌ＿ｃｏｄｉｎｇ＿ｒｉｃｅ＿ｃｏｎ

straint_flagが存在しない場合、gci_no_ts_residual_coding_rice_constraint_flagの値が0に等しいと推測する。

【0097】

gci_no_rrc_rice_extension_constraint_flagが1に等しい場合、OlsInScope内のすべての画像のsps_rrc_rice_extension_flagが0に等しくなることを指定する。gci_no_rrc_rice_extension_constraint_flagが0に等しい場合、このような制約は課されない。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0に等しいと推測する。

【0098】

gci_no_persistent_rice_adaptation_constraint_flagが1に等しい場合、OlsInScope内のすべての画像のsps_persistent_rice_adaptation_enabled_flagが0に等しくなることを指定する。gci_no_persistent_rice_adaptation_constraint_flagが0に等しい場合、このような制約は課されない。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0に等しいと推測する。

【0099】

gci_no_reverse_last_sig_coeff_constraint_flagが1に等しい場合、OlsInScope内のすべての画像のsps_reverse_last_sig_coeff_enabled_flagが0に等しくなることを指定する。gci_no_reverse_last_sig_coeff_constraint_flagが0に等しい場合、このような制約は課されない。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0に等しいと推測する。

【0100】

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。

【0101】

gci_num_additional_bitsは、gci_alignment_zero_bit構文要素（存在する場合）以外の一般制約情報構文構造内の追加ＧＣＩビットの数を指定する。当該ドキュメントのこの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≫6より大きい値は、ＩＴＵ－Ｔ｜ＩＳＯ／ＩＥＣによる将来の使用のために予約されている。当該ドキュメントのこのバージョンでは、gci_num_additional_bitsの値が0又は≪1≫6に等しくなるように要求されているが、当該ドキュメントのこのバージョンに準拠するデコーダは、構文に≪1≫6より大きいgci_num_additional_bitsの値が表示されることを許可し、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、追加ＧＣＩビットによって指定されたすべての制約フラグが0に等しいと推測する。

【0102】

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。

【０１０３】

ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓは、ｇｃｉ＿ａｌｉｇｎｍｅｎｔ＿ｚｅｒｏ＿ｂｉｔ構文要素（存在する場合）以外の一般制約情報構文構造内の追加ＧＣＩビットの数を指定する。当該ドキュメントのこの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≫６より大きい値は、ＩＴＵ－Ｔ｜ＩＳＯ／ＩＥＣによる将来の使用のために予約されている。当該ドキュメントのこのバージョンでは、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値が０又は≪１≫６に等しくなるように要求されているが、当該ドキュメントのこのバージョンに準拠するデコーダは、構文に≪１≫６より大きいｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値が表示されることを許可し、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、追加ＧＣＩビットによって指定されたすべての制約フラグが０に等しいと推測する。

【０１０４】

別の例では、一般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。

【０１０５】

【表６】

general_constraints_info( ) {	**記述子**
gci_present_flag	u(1)
if(gci_present_flag) {	
/* general */	
gci_intra_only_constraint_flag	u(1)
...	
gci_num_additional_bits	u(8)
if(gci_num_additional_bits > ~~0~~ 5) {	
additional_general_constraints_info( )	
~~**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)
}	

【０１０６】

【表７】

additional_general_constraints_info( ) {	記述子
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)
}	

【０１０７】

ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓは、ｇｃｉ＿ａｌｉｇｎｍｅｎｔ＿ｚｅｒｏ＿ｂｉｔ構文要素（存在する場合）以外の一般制約情報構文構造内の追加ＧＣＩビットの数を指定する。当該ドキュメントのこの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≫６より大きい値は、ＩＴＵ－Ｔ｜ＩＳＯ／ＩＥＣによる将来の使用のために予約されている。当該ドキュメントのこのバージョンでは、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値が０又は≪１≫６に等しくなるように要求されているが、当該ドキュメントのこのバージョンに準拠するデコーダは、構文に≪１≫６より大きいｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値が表示されることを許可し、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＿ｃｏｎｓｔｒａｉｎｔｓ＿ｉｎｆｏ（）構文構造は、いかなる制約も課さない。

【０１０８】

別の例では、一般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。

【０１０９】

ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓは、ｇｃｉ＿ａｌｉｇｎｍｅｎｔ＿ｚｅｒｏ＿ｂｉｔ構文要素（存在する場合）以外の一般制約情報構文構造内の追加ＧＣＩビットの数を指定する。当該ドキュメントのこの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≫６より大きい値は、ＩＴＵ－Ｔ｜ＩＳＯ／ＩＥＣによる将来の使用のために予約されている。当該ドキュメントのこのバージョンでは、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値が０又は≪１≫６に等しくなるように要求されているが、当該ドキュメントのこのバージョンに準拠するデコーダは、構文に≪１≫６より大きいｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値が表示されることを許可し、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＿ｃｏｎｓｔｒａｉｎｔｓ＿ｉｎｆｏ（）構文構造は、いかなる制約も課さない。

【０１１０】

一般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、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は、これらのフラグの各セマンティックによって指定された対応する制約を常に課すことができる。言い換えると、ＧＣＩフラグが存在し、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓが０に等しい場合、これら６つのＧＣＩフラグが１に等しいと推測する。

【０１１１】

一例として、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓのセマンティックに対する可能な変更は、次に示す通りであり、これは、ＶＶＣバージョン２仕様の現在の追加に基づいて行われる。

【０１１２】

ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓは、ｇｃｉ＿ａｌｉｇｎｍｅｎｔ＿ｚｅｒｏ＿ｂｉｔ構文要素（存在する場合）以外の一般制約情報構文構造内の追加ＧＣＩビットの数を指定する。当該ドキュメントのこの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。ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの１より大きい値は、ＩＴＵ－Ｔ｜ＩＳＯ／ＩＥＣによる将来の使用のために予約されている。当該ドキュメントのこのバージョンでは、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値が０又は１に等しくなるように要求されているが、当該ドキュメントのこのバージョンに準拠するデコーダは、構文に１より大きいｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値が表示されることを許可し、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、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。

【０１１３】

別の例として、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓのセマンティックに対する可能な変更は、次に示す通りであり、これは、ＶＶＣバージョン２仕様の現在の追加に基づいて行われる。

【０１１４】

ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓは、ｇｃｉ＿ａｌｉｇｎｍｅｎｔ＿ｚｅｒｏ＿ｂｉｔ構文要素（存在する場合）以外の一般制約情報構文構造内の追加ＧＣＩビットの数を指定する。当該ドキュメントのこの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又は６以外の値は、ＩＴＵ－Ｔ｜ＩＳＯ／ＩＥＣによる将来の使用のために予約されている。当該ドキュメントのこのバージョンでは、ｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値が０又は≪１≫６に等しくなるように要求されているが、当該ドキュメントのこのバージョンに準拠するデコーダは、構文に≪１より大きい≫０又は６以外のｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓの値が表示されることを許可し、且つｇｃｉ＿ｎｕｍ＿ａｄｄｉｔｉｏｎａｌ＿ｂｉｔｓが≪１より大きい≫０又は６以外である場合は、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、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。

【０１１５】

上述の実施例に加えて、又は上述の実施例の代替として、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。

【０１１６】

【表８】

general_constraints_info( ) {	記述子
gci_present_flag	u(1)
if(gci_present_flag) {	
/* 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)
}	

【０１１７】

≪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指定された復号化プロセスには影響しない。当該仕様のこの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。

【０１１８】

さらなる実施例では、構文要素ｇｃｉ＿ｒｅｓｅｒｖｅｄ＿ｚｅｒｏ＿ｂｉｔ［ｉ］の名称がｇｃｉ＿ｒｅｓｅｒｖｅｄ＿ｂｉｔ［ｉ］に変更される。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。

【０１１９】

【表９】

general_constraints_info( ) {	記述子
gci_present_flag	u(1)
if(gci_present_flag) {	
/* 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)
}	

【０１２０】

≪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指定された復号化プロセスには影響しない。当該仕様のこの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。

【０１２１】

一般制約フラグについて

ｇｃｉ＿ａｌｌ＿ｒａｐ＿ｐｉｃｔｕｒｅｓ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇ

ｇｃｉ＿ａｌｌ＿ｒａｐ＿ｐｉｃｔｕｒｅｓ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇは、画像がＩＲＡＰ又はＧＤＲ画像に制限されることを示すために使用される。

【０１２２】

ＮＡＬ（Ｎｅｔｗｏｒｋ　Ａｂｓｔｒａｃｔｉｏｎ　Ｌａｙｅｒ）は、ＶＶＣ構文要素を「ＮＡＬユニット」に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、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）に合わせて、ＶＶＣを簡単かつ効果的にカスタマイズできる。ＶＶＣ規格における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。

【０１２３】

【表１０（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	後続画像又はサブ画像の符号化スライス＊ｓｌｉｃｅ＿＿ｌａｙｅｒ＿＿ｒｂｓｐ（）	ＶＣＬ
１	ＳＴＳＡ＿＿ＮＵＴ	ＳＴＳＡ画像又はサブ画像の符号化スライス＊ｓｌｉｃｅ＿＿ｌａｙｅｒ＿＿ｒｂｓｐ（）	ＶＣＬ
２	ＲＡＤＬ＿＿ＮＵＴ	ＲＡＤＬ画像又はサブ画像の符号化スライス＊ｓｌｉｃｅ＿＿ｌａｙｅｒ＿＿ｒｂｓｐ（）	ＶＣＬ
３	ＲＡＳＬ＿＿ＮＵＴ	ＲＡＳＬ画像又はサブ画像の符号化スライス＊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
７８	ＩＤＲ＿＿Ｗ＿＿ＲＡＤＬＩＤＲ＿＿Ｎ＿＿ＬＰ	ＩＤＲ画像又はサブ画像の符号化スライス＊ｓｌｉｃｅ＿＿ｌａｙｅｒ＿＿ｒｂｓｐ（）	ＶＣＬ
９	ＣＲＡ＿＿ＮＵＴ	ＣＲＡ画像又はサブ画像の符号化スライス＊ｓｌｉｃｅ＿＿ｌａｙｅｒ＿＿ｒｂｓｐ（）	ＶＣＬ
１０	ＧＤＲ＿＿ＮＵＴ	ＧＤＲ画像又はサブ画像の符号化スライス＊ｓｌｉｃｅ＿＿ｌａｙｅｒ＿＿ｒｂｓｐ（）	ＶＣＬ
１１	ＲＳＶ＿＿ＩＲＡＰ＿＿１１	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＿＿ｒｂｓｐ（）	非ＶＣＬ
１３	ＤＣＩ＿＿ＮＵＴ	復号化能力情報ｄｅｃｏｄｉｎｇ＿＿ｃａｐａｂｉｌ	非ＶＣＬ

【表１０（２）】

		ｉｔｙ＿ｉｎｆｏｒｍａｔｉｏｎ＿ｒｂｓｐ（）	
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	画像パラメータセット 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	画像ヘッダー 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
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＿２６ ＲＳＶ＿ＮＶＣＬ＿２７	予約済みの非ＶＣＬ　ＮＡＬユニットタイプ	非ＶＣＬ
２８．．３１	ＵＮＳＰＥＣ＿２８．．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に等しい場合の画像の属性、及びｐｐｓ＿ｍｉｘｅｄ＿ｎａｌｕ＿ｔｙｐｅｓ＿ｉｎ＿ｐｉｃ＿ｆｌａｇが１に等しい場合のサブ画像の属性を示す。			

【０１２４】

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画像は、ＶＣＬ　ＮＡＬユニットに基づいて復号化される。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のに役立つ。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、インター予測ツールは、以前に復号化された（参照）画像へのアクセスに依存する。イントラ予測ツールのみを使用して圧縮された画像と比較して、インター予測ツールを使用して符号化された画像は、より効率的に圧縮されることができる。

しかし、参照画像が利用できない場合では、このような復号化依存性により問題が発生する。

【０１２５】

イントラランダムアクセスポイント（ＩＲＡＰ）画像は、符号化された画像であり、そのすべてのＶＣＬ　ＮＡＬユニットは、ＩＤＲ＿Ｗ＿ＲＡＤＬからＣＲＡ＿ＮＵＴまでの範囲（端点を含む）内で同じｎａｌ＿ｕｎｉｔ＿ｔｙｐｅ値を有する。ＩＲＡＰ画像は、ＣＲＡ画像又はＩＤＲ画像であってもよい。ＩＲＡＰ画像は、復号化プロセスで同じレイヤの参照画像からのインター予測を使用しない。復号化順でビットストリームにおける１番目の画像は、ＩＲＡＰ画像又はＧＤＲ（Ｇｒａｄｕａｌ　ｄｅｃｏｄｅｒ　ｒｅｆｒｅｓｈ）画像である。シングルレイヤービットストリームの場合、必要なパラメータセットが、参照される必要があるときに利用可能であれば、復号化順でＩＲＡＰ画像に先行する画像に対して復号化プロセスを実行せずに、ＩＲＡＰ画像と、ＣＬＶＳにおいて復号化順で配列されたすべての後続の非ＲＡＳＬ画像を正しく復号化することができる。ＩＲＡＰ画像のｐｐｓ＿ｍｉｘｅｄ＿ｎａｌｕ＿ｔｙｐｅｓ＿ｉｎ＿ｐｉｃ＿ｆｌａｇの値は０に等しい。ある画像のｐｐｓ＿ｍｉｘｅｄ＿ｎａｌｕ＿ｔｙｐｅｓ＿ｉｎ＿ｐｉｃ＿ｆｌａｇが０に等しく、且つ当該画像の任意のスライス（ｓｌｉｃｅ）のｎａｌ＿ｕｎｉｔ＿ｔｙｐｅが、ＩＤＲ＿Ｗ＿ＲＡＤＬからＣＲＡ＿ＮＵＴまでの範囲（端点を含む）内にある場合、当該画像の他のすべてのスライスは同じｎａｌ＿ｕｎｉｔ＿ｔｙｐｅ値を有し、最初のスライスを受信した後、当該画像はＩＲＡＰ画像であることが分かる。

【０１２６】

したがって、ＩＲＡＰ画像は、同じレイヤ間でインター予測を使用しない。当該制限により、ＩＲＡＰ画像は、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場所の検索ポイントとして使用できる。しかし、ＩＲＡＰ画像の圧縮効率は、通常、非ＩＲＡＰ画像の圧縮効率より低い。

【０１２７】

ＧＤＲ（Ｇｒａｄｕａｌ　ｄｅｃｏｄｅｒ　ｒｅｆｒｅｓｈ）画像は、非ＩＲＡＰ画像とＩＲＡＰ画像との間のトレードオフ（ｔｒａｄｅ－ｏｆｆ）としてＶＶＣ規格で導入された。ＧＤＲ画像は、インター予測を使用しない「クリーン（ｃｌｅａｎ）」な部分があり、画像の残りの部分は、インター予測を自由に使用できる。このように画像を分割することで、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化される。連続するＧＤＲ画像では、「クリーン」な部分の空間位置が回転するため、最終的には画像全体がエラーから回復することができる。

【０１２８】

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、すべての画像をＩＲＡＰ画像又はＧＤＲ画像に制限することが望ましい場合がある。ＶＶＣｖ２では、ＧＣＩ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ようになった。ｇｃｉ＿ａｌｌ＿ｒａｐ＿ｐｉｃｔｕｒｅｓ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが１に等しい場合、ＯｌｓＩｎＳｃｏｐｅ内のすべての画像がＩＲＡＰ画像、又はｐｈ＿ｒｅｃｏｖｅｒｙ＿ｐｏｃ＿ｃｎｔが０に等しいＧＤＲ画像であることを指定する。ｇｃｉ＿ａｌｌ＿ｒａｐ＿ｐｉｃｔｕｒｅｓ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが０に等しい場合、このような制約は課されない。ｇｃｉ＿ａｌｌ＿ｒａｐ＿ｐｉｃｔｕｒｅｓ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが存在しない場合、ｇｃｉ＿ａｌｌ＿ｒａｐ＿ｐｉｃｔｕｒｅｓ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇの値が０に等しいと推測する。

【０１２９】

ｐｒｏｆｉｌｅ＿ｔｉｅｒ＿ｌｅｖｅｌ（）構文構造がＶＰＳに含まれている場合、ＯｌｓＩｎＳｃｏｐｅは、ＶＰＳによって指定された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。

【０１３０】
ｇｃｉ＿ｎｏ＿ｅｘｔｅｎｄｅｄ＿ｐｒｅｃｉｓｉｏｎ＿ｐｒｏｃｅｓｓｉｎｇ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇ
ＧＣＩフラグｇｃｉ＿ｎｏ＿ｅｘｔｅｎｄｅｄ＿ｐｒｅｃｉｓｉｏｎ＿ｐｒｏｃｅｓｓｉｎｇ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇは、拡張変換精度のＶＶＣｖ２ツールが制約されているか否かを高レベルでシグナリングする。ＶＶＣ規格では、ビデオ信号の画素は、整数値で表される。ＶＶＣ規格で説明されたすべての計算と処理は、整数演算で表現される。この制限は、複雑さと相互運用性の両方の理由から重要である。第１に、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。第２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、必ずしも交換可能ではなく（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間での浮動小数点演算の評価が同じであるとは保証されない。
【０１３１】
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規格では、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。元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、これらの残差係数は、拡張されたビット深度（例えば、ＢｉｔＤｅｐｔｈ＋１）を有する可能性があるが、ＶＶＣ規格では、残差係数は、ＢｉｔＤｅｐｔｈのビット深度を維持するためにクリップされる。実際には、実際のエンコーダは、元のビデオ信号より大きいマグニチュード（ｍａｇｎｉｔｕｄｅ）の残差を生成する予測ツールを選択しないため、最悪のケースが発生しない。
【０１３２】
その後、残差係数は通常、整数化された離散コサイン変換（ＤＣＴ）によって分析され、変換係数が生成される。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。
【０１３３】
【数１】
$$f: \mathbb{R}^N \to \mathbb{R}^N$$

ここで、
【数２】
$$\mathbb{R}^N$$

は、実数のＮ次元空間を表す。ＶＶＣ規格では、ＤＣＴの整数化された近似（即ち、
【数３】
$$f: \mathbb{Z}^N \to \mathbb{Z}^N$$

）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上の決定である。
【０１３４】
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。
【０１３５】

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、（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変換係数に対して拡張ダイナミックレンジが使用されること、及びａｂｓ＿ｒｅｍａｉｎｉｎｇ［］とｄｅｃ＿ａｂｓ＿ｌｅｖｅｌ［］構文要素の二値化に対して拡張ダイナミックレンジを使用することを指定する。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対して拡張ダイナミックレンジが使用されないことを指定する。存在しない場合、ｓｐｓ＿ｅｘｔｅｎｄｅｄ＿ｐｒｅｃｉｓｉｏｎ＿ｆｌａｇの値が０に等しいと推測する。変数Ｌｏｇ２ＴｒａｎｓｆｏｒｍＲａｎｇｅは、次のように導出される：

Ｌｏｇ２ＴｒａｎｓｆｏｒｍＲａｎｇｅ＝ｓｐｓ＿ｅｘｔｅｎｄｅｄ＿ｐｒｅｃｉｓｉｏｎ＿ｆｌａｇ？Ｍａｘ（１５，Ｍｉｎ（２０，ＢｉｔＤｅｐｔｈ＋６））：１５

ＣｏｅｆｆＭｉｎ＝－（１＜＜（ｓｐｓ＿ｅｘｔｅｎｄｅｄ＿ｐｒｅｃｉｓｉｏｎ＿ｆｌａｇ？Ｍａｘ（１５，Ｍｉｎ（２０，ＢｉｔＤｅｐｔｈ＋６））：１５））

ＣｏｅｆｆＭａｘ＝（１＜＜（ｓｐｓ＿ｅｘｔｅｎｄｅｄ＿ｐｒｅｃｉｓｉｏｎ＿ｆｌａｇ？Ｍａｘ（１５，Ｍｉｎ（２０，ＢｉｔＤｅｐｔｈ＋６））：１５））－１

ｇｃｉ＿ｎｏ＿ｅｘｔｅｎｄｅｄ＿ｐｒｅｃｉｓｉｏｎ＿ｐｒｏｃｅｓｓｉｎｇ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが１に等しい場合、ＯｌｓＩｎＳｃｏｐｅ内のすべての画像のｓｐｓ＿ｅｘｔｅｎｄｅｄ＿ｐｒｅｃｉｓｉｏｎ＿ｆｌａｇが０に等しくなることを指定する。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によって指定される１つ又は複数の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、このような制約は課されない。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０に等しいと推測する。

【０１３６】

ｇｃｉ＿ｎｏ＿ｔｓ＿ｒｅｓｉｄｕａｌ＿ｃｏｄｉｎｇ＿ｒｉｃｅ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇ

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シグナリングのＶＶＣｖ２ツールが制約されているか否かを高レベルでシグナリングする。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、ビットストリームに挿入されるビットシーケンスに符号化される。

【０１３７】

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エンジンは、適応性を有し、構文要素を理論的なシャノン限界（Ｓｈａｎｎｏｎ　ｌｉｍｉｔ）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、０又は１を超える値を持つ構文要素）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に対応するｂｉｎが存在しないことを示す。

【０１３８】

【表１１】

表：固定長の二値化の例

値	二値化
０	００
１	０１
２	１０
３	１１

【０１３９】

【表１２】

表：可変長の二値化の例

値	二値化
０	０
１	１０
２	１１０
３	１１１０
４	１１１１

【０１４０】

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。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には個別のコンテキストが必要である。

【０１４１】

ＣＡＢＡＣのみで、値の範囲が広い構文要素を符号化することは理想的ではなく、なぜなら、このような構文要素は二値化が長くなるため、コンテキストの記憶と更新に関して過度のオーバーヘッドが発生するためである。例えば、ＣＡＢＡＣのみで残差係数値を符号化することは不便である。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。

【０１４２】

ライスコーディングは、ゴロムコーディング（Ｇｏｌｏｍｂ　ｃｏｄｉｎｇ）、又はライス－ゴロムコーディング（Ｒｉｃｅ－Ｇｏｌｏｍｂ　ｃｏｄｉｎｇ）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ダであり、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、エントロピコーダは、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、ゴロムコーダと同等であり、ここで、$M = 2^R$である。

【０１４３】

非負整数値ｘは、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される。商ｑと剰余ｒは、次のように計算される。

【０１４４】

【数４】

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

$$r = x \mod 2^{R}$$

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）の連結である。プレフィックスコードは、ｑの単項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、切捨単項コード（Ｔｒｕｎｃａｔｅｄ　Ｕｎａｒｙ　Ｃｏｄｅ）である。

【０１４５】

【表１３】

ｑ	単項コード
０	０
１	１０
２	１１０
３	１１１０
４	１１１１０
５	１１１１１０
６	１１１１１１

【０１４６】

単項コードが切り捨てられると、ライスコードも切り捨てられ、これは、符号化可能なｘの値の範囲は有限であることを意味する。ＶＶＣでは、切捨ライスコード（Ｔｒｕｎｃａｔｅｄ　Ｒｉｃｅ　Ｃｏｄｅ）は、［０，６＊２^{R}］の範囲のｘの値に適用される。

【０１４７】

サフィックスコードは、ｒを二値化した固定長コードであり、Ｒビットを有する。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。

【０１４８】

【表１４】

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

【０１４９】

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。小さいマグニチュードの残差の値をシグナリングするのに十分な、少数の構文要素フラグが定義されている。例えば、ｓｉｇ＿ｃｏｅｆｆ＿ｆｌａｇは、残差のマグニチュードがゼロか非ゼロかを示す。ｓｉｇ＿ｃｏｅｆｆ＿ｆｌａｇが１である場合、さらにフラグをシグナリングすることができ、フラグは、通常、ａｂｓ＿ｌｅｖｅｌ＿ｇｔｘ＿ｆｌａｇと命名され、残差のマグニチュードが１、２、３などより大きいか否かを示す。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、残差のほとんどは、ＣＡＢＡＣによって、比較的に少数のｂｉｎを使用して効率的に符号化できる。

【０１５０】

残差係数フラグによってシグナリング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でシグナリングされる。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シグナリングされる。ａｂｓ＿ｒｅｍａｉｎｄｅｒの値が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（ａｂｓ＿ｒｅｍａｉｎｄｅｒ－６＊２^{R}）の指数ゴロムコードの連結によってシグナリングされる。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。

【０１５１】

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。残差係数の統計に合わせるために、ライスパラメータは、ｌｏｃＳｕｍＡｂｓ値に基づいて適応的に決定され、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。

【０１５２】

【表１５】

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

【０１５３】

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：Ｒｅｇｕｌａｒ　Ｒｅｓｉｄｕａｌ　Ｃｏｄｉｎｇ）における残差係数のために設計され、ＲＲＣにおける残差係数は、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、適応ライスパラメータの決定は、「変換スキップ残差コーディング」（ＴＳＲＣ：Ｔｒａｎｓｆｏｒｍ－Ｓｋｉｐ　Ｒｅｓｉｄｕａｌ　Ｃｏｄｉｎｇ）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が認識された。

【０１５４】

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＿ｍｉｎｕｓｌによってライスパラメータを明示的にシグナリングすることを許可する。当該構文要素がシグナリングされると、ライスパラメータは、Ｒ＝「ｓｈ＿ｔｓ＿ｒｅｓｉｄｕａｌ＿ｃｏｄｉｎｇ＿ｒｉｓｅ＿ｉｄｘ＿ｍｉｎｕｓｌ」＋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存在し続ける。

【０１５５】

【表１６】

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)

【０１５６】

ｓｈ＿ｔｓ＿ｒｅｓｉｄｕａｌ＿ｃｏｄｉｎｇ＿ｒｉｃｅ＿ｉｄｘ＿ｍｉｎｕｓｌプラス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指定する。存在しない場合、ｓｈ＿ｔｓ＿ｒｅｓｉｄｕａｌ＿ｃｏｄｉｎｇ＿ｒｉｃｅ＿ｉｄｘ＿ｍｉｎｕｓｌの値が０に等しいと推測する。

【０１５７】

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＿ｒｅｓｉ

dual_coding_rice_present_in_sh_flagによって制御される。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シグナリングは有効にならない。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シグナリングのVVCv2ツールが制約されているか否かを高レベルでシグナリングする。gci_no_ts_residual_coding_rice_constraint_flagが1に等しい場合、OlsInScope内のすべての画像のsps_ts_residual_coding_rice_present_in_sh_flagが0に等しくなることを指定する。gci_no_ts_residual_coding_rice_constraint_flagが0に等しい場合、このような制約は課されない。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0に等しいと推測する。

【0158】

gci_no_rrc_rice_extension_constraint_flag

gci_no_rrc_rice_extension_constraint_flagは、OlsInScope内のすべての画像のsps_rrc_rice_extension_flagを指定する。gci_no_rrc_rice_extension_constraint_flagが1に等しい場合、OlsInScope内のすべての画像のsps_rrc_rice_extension_flagが0に等しくなることを指定する。gci_no_rrc_rice_extension_constraint_flagが0に等しい場合、このような制約は課されない。

【0159】

高ビット深度と高ビットレートのアプリケーションでは、RRCの多くの位置に対して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、ライスパラメータを大きくすると、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、代替ライスパラメータ導出は、sps_rrc_rice_extension_flagによってシグナリングできる。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指定する。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指定する。存在しない場合、sps_rrc_rice_extension_flagの値が0に等しいと推測する。

【0160】

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と左上隅の輝度位置（x0，y0）を持つ変換ブロックの配列AbsLevel[x][y]が与えられると、変数locSumAbsは、以下の擬似コードプロセスで指定されたように導出される（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）。

【0161】

```
locSumAbs＝0
if（xC＜（1＜＜AbsL≪l≫og2TbWidth）－1）{
  locSumAbs＋＝AbsLevel[xC＋1][yC]
  if（xC＜（1＜＜AbsL≪l≫og2TbWidth）－2）
    locSumAbs＋＝AbsLevel[xC＋2][yC]
```

else
locSumAbs+=HistValue
if(yC<(1<<AbsL≪l≫og2TbHeight)-1)
locSumAbs+=AbsLevel[xC+1][yC+1] (1517)
else
locSumAbs+=HistValue
} else
locSumAbs+=2*HistValue
if(yC<(1<<AbsL≪l≫og2TbHeight)-1) {
locSumAbs+=AbsLevel[xC][yC+1]
if(yC<(1<<AbsL≪l≫og2TbHeight)-2)
locSumAbs+=AbsLevel[xC][yC+2]
else
locSumAbs+=HistValue
} else
locSumAbs+=HistValue
リストTx[]とRx[]は、次のように指定される：
Tx[]={32,128,512,2048} (1523)
Rx[]={0,2,4,6,8} (1524)
変数shiftValの値は、次のように導出される：
if(!sps_rrc_rice_extension_flag)
shiftVal=0 (1526X1)
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：
locSumAbs=Clip3(0,31,(locSumAbs>>shiftVal)-baseLevel*5) (1526X2)
変数locSumAbsが与えられると、ライスパラメータcRiceParamは表128で指定されているように導出され、次に、cRiceParamは次のように更新される：
cRiceParam=cRiceParam+shiftVal (1526X3)
baseLevelが0に等しい場合、変数ZeroPos[n]は、次のように導出される：
ZeroPos[n]=(Qstate<2?1:2)<<cRiceParam (1518)
【0162】

【表１７】

表１２８－ｌｏｃＳｕｍＡｂｓに基づくｃＲｉｃｅＰａｒａｍの説明

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

【０１６３】

（１５２６Ｘ３）から、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、ＶＶＣｖ１と比較してＶＶＣｖ２では大きくなる可能性があることが分かる。

【０１６４】

ｇｃｉ＿ｎｏ＿ｐｅｒｓｉｓｔｅｎｔ＿ｒｉｃｅ＿ａｄａｐｔａｔｉｏｎ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇ

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導出に対する制約をシグナリングする。ｇｃｉ＿ｎｏ＿ｐｅｒｓｉｓｔｅｎｔ＿ｒｉｃｅ＿ａｄａｐｔａｔｉｏｎ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが１に等しい場合、ＯｌｓＩｎＳｃｏｐｅ内のすべての画像のｓｐｓ＿ｐｅｒｓｉｓｔｅｎｔ＿ｒｉｃｅ＿ａｄａｐｔａｔｉｏｎ＿ｅｎａｂｌｅｄ＿ｆｌａｇが０に等しくなることを指定する。ｇｃｉ＿ｎｏ＿ｐｅｒｓｉｓｔｅｎｔ＿ｒｉｃｅ＿ａｄａｐｔａｔｉｏｎ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが０に等しい場合、このような制約は課されない。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０に等しいと推測する。

【０１６５】

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指定する。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示す。存在しない場合、ｓｐｓ＿ｐｅｒｓｉｓｔｅｎｔ＿ｒｉｃｅ＿ａｄａｐｔａｔｉｏｎ＿ｅｎａｂｌｅｄ＿ｆｌａｇの値が０に等しいと推測する。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）。

【０１６６】

ＣＴＵがスライス（ｓｌｉｃｅ）又はタイル（ｔｉｌｅ）内の最初のＣＴＵである場合、サブ条項９．３．２．２で指定されたようにコンテキスト変数の初期化プロセスが呼び出され、配列ＰｒｅｄｉｃｔｏｒＰａｌｅｔｔｅＳｉｚｅ［ｃｈＴｙｐｅ］（ここで、ｃｈＴｙｐｅ＝０、１）は０に初期化され、配列ＳｔａｔＣｏｅｆｆ［ｉ］（ここで、ｉ＝０．．２）は次のように初期化される：

ＳｔａｔＣｏｅｆｆ［ｉ］＝ｓｐｓ＿ｐｅｒｓｉｓｔｅｎｔ＿ｒｉｃｅ＿ａｄａｐｔａｔｉｏｎ＿ｅｎａｂｌｅｄ＿ｆｌａｇ？２＊Ｆｌｏｏｒ（Ｌｏｇ２（ＢｉｔＤｅｐｔｈ－１０））：０　（１５１３Ｘ３）

ＳｔａｔＣｏｅｆｆ［ｉ］は、ＨｉｓＶａｌｕｅを計算するために使用され、ＨｉｓＶａｌｕｅは、（１５１７）のｌｏｃＳｕｍＡｂｓを計算するために使用され、ＴＵごとに１回更新されることができる。ブロック境界にあるこれらの位置のライスパラメータの導出は、ＨｉｓＶａｌｕｅの支援によって正確な値を有し得る。

【０１６７】

ｇｃｉ＿ｎｏ＿ｒｅｖｅｒｓｅ＿ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇ

ｇｃｉ＿ｎｏ＿ｒｅｖｅｒｓｅ＿ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが１に等しい場合、ＯｌｓＩｎＳｃｏｐｅ内のすべての画像のｓｐｓ＿ｒｅｖｅｒｓｅ＿ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｅｎａｂｌｅｄ＿ｆｌａｇが０に等しくなることを指定する。ｇｃｉ＿ｎｏ＿ｒｅｖｅｒｓｅ＿ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが０に等しい場合、このような制約は課されない。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０に等しいと推測する。

【０１６８】

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。当該位置は、ＶＶＣｖ１において現在ＴＵの（ｘ，ｙ）と（０，０）との差を使用して符号化される。これは、ＶＶＣｖ１では妥当であり、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、これは、ＶＶＣｖ２アプリケーションには当てはまらない可能性があり、多くの非ゼロレベルがＴＵ全体に広がるため、この位置（ｘ，ｙ）を左上隅（０，０）ではなく右下隅を基準として符号化する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。

【０１６９】

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、最後の有効係数（ｌａｓｔ　ｓｉｇｎｉｆｉｃａｎｔ　ｃｏｅｆｆｉｃｉｅｎｔ）の座標は（（Ｌｏｇ２ＺｏＴｂＷｉｄｔｈ＜＜１）－１、（Ｌｏｇ２ＺｏＴｂＨｅｉｇｈｔ＜＜１）－１）を基準として符号化されることを指定する。ｓｈ＿ｒｅｖｅｒｓｅ＿ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｆｌａｇが０に等しい場合、最後の有効係数の座標が現在スライスの各変換ブロックの（０，０）を基準として符号化されることを指定する。存在しない場合、ｓｈ＿ｒｅｖｅｒｓｅ＿ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｆｌａｇの値が０に等しいと推測する。

【０１７０】

ｓｈ＿ｒｅｖｅｒｓｅ＿ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｆｌａｇは、スライスヘッダー内のｓｐｓ＿ｒｅｖｅｒｓｅ＿ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｅｎａｂｌｅｄ＿ｆｌａｇと条件付きで次のように解析される。

【０１７１】

－ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｘ＿ｓｕｆｆｉｘが存在しない場合、次のことが適用される：

ＬａｓｔＳｉｇｎｉｆｉｃａｎｔＣｏｅｆｆＸ＝ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｘ＿ｐｒｅｆｉｘ　　（１９３）

－そうでない場合（ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｘ＿ｓｕｆｆｉｘが存在する）、次のことが適用される：

ＬａｓｔＳｉｇｎｉｆｉｃａｎｔＣｏｅｆｆＸ＝（１＜＜（（ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｘ＿ｐｒｅｆｉｘ＞＞１）－１））＊　（１９４）

（２＋（ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｘ＿ｐｒｅｆｉｘ＆１））＋ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｘ＿ｓｕｆｆｉｘ
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：
ＬａｓｔＳｉｇｎｉｆｉｃａｎｔＣｏｅｆｆＸ＝（１＜＜Ｌｏｇ２ＺｏＴｂＷｉｄｔｈ）－１－ＬａｓｔＳｉｇｎｉｆｉｃａｎｔＣｏｅｆｆＸ　（１９５）
－ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｙ＿ｓｕｆｆｉｘが存在しない場合、次のことが適用される：
ＬａｓｔＳｉｇｎｉｆｉｃａｎｔＣｏｅｆｆＹ＝ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｙ＿ｐｒｅｆｉｘ　（１９６）
－そうでない場合（ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｙ＿ｓｕｆｆｉｘが存在する）、次のことが適用される：
ＬａｓｔＳｉｇｎｉｆｉｃａｎｔＣｏｅｆｆＹ＝（１＜＜（（ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｙ＿ｐｒｅｆｉｘ＞＞１）－１））＊　（１９７）
（２＋（ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｙ＿ｐｒｅｆｉｘ＆１））＋ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｙ＿ｓｕｆｆｉｘ
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：
ＬａｓｔＳｉｇｎｉｆｉｃａｎｔＣｏｅｆｆＹ＝（１＜＜Ｌｏｇ２ＺｏＴｂＨｅｉｇｈｔ）－１－ＬａｓｔＳｉｇｎｉｆｉｃａｎｔＣｏｅｆｆＹ　（１９８）

【０１７２】

ＶＶＣｖ２スライスヘッダー

【表１８】

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)

【０１７３】

図５は、本発明のいくつかの実施例に係る、ビデオ復号化プロセス５００の一例を示す。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５に示す動作を実現することができ、当該コンピューティング機器は、例えば、エントロピ復号化モジュール２１６、逆量子化モジュール２１８、逆変換モジュール２１９、インループ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５００について説明する。ただし、他の実現方式も可能である。

【０１７４】

ブロック５０２において、プロセス５００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。ブロック５０４において

、プロセス５００は、ビデオのビットストリームから一般制約情報（ＧＣＩ）フラグを抽出することを含む。前述のように、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指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、追加符号化ツールに課されている制約を示すために使用されることを指定する。ｇｃｉ＿ｐｒｅｓｅｎｔ＿ｆｌａｇが０に等しい場合、ＧＣＩ構文要素が存在せず、ビデオに一般制約が課されないことを指定する。エンコーダによっては、ＧＣＩフラグは、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抽出されることができる。

【０１７５】

ブロック５０６において、プロセス５００は、ＧＣＩフラグの値に基づいて、ビデオに１つ又は複数の一般制約が課されているか否かを決定することを含む。課されている場合（即ち、ＧＣＩフラグが１である）、ブロック５０８において、プロセス５００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、複数のフラグビットを含み、当該複数のフラグビットは、ビデオに対して制約される対応する追加符号化ツールを示す。

【０１７６】

ブロック５１０において、プロセス５００は、Ｍ＞５であるか否かを決定することを含む。そうである場合、ブロック５１２において、プロセス５００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、対応するフラグを表し、これらの対応するフラグは、６つの追加符号化ツールの対応する制約を示す。これら６つのフラグは、ビデオの画像がイントラランダムアクセスポイント（ＩＲＡＰ）画像又は漸次復号リフレッシュ（ＧＤＲ）画像に制限されることを示すフラグｇｃｉ＿ａｌｌ＿ｒａｐ＿ｐｉｃｔｕｒｅｓ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇと、拡張変換精度が制約され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シグナリングが制約され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いて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化するときに、ＯｌｓＩｎＳｃｏｐｅ内の画像に制約を課すか否かを示すフラグｇｃｉ＿ｎｏ＿ｒｅｖｅｒｓｅ＿ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇと、を含む。

【０１７７】

Ｍが６より大きい場合、ブロック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、このＭ－６ビットを破棄することを含む。つまり、ビデオの復号化は、このＭ－６ビットとは独立して実行される。

【０１７８】

ブロック５１４において、プロセス５００は、これら６つのフラグによって示される６つの追加符号化ツールの制約に基づいて、ビデオのビットストリームの残りの部分を画像に復号化することを含む。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画像がＩＲＡＰ画像又はｐｈ＿ｒｅｃｏｖｅｒｙ＿ｐｏｃ＿ｃｎｔが０に等しいＧＤＲ画像であると決定し、前記１つ又は複数の出力レイヤセット内のＧＤＲ画像又はＩＲＡＰ画像を復号化することができる。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、ＯｌｓＩｎＳｃｏｐｅ内の画像のｓｐｓ＿ｅｘｔｅｎｄｅｄ＿ｐｒｅｃｉｓｉｏｎ＿ｆｌａｇを０に設定し

て、拡張ダイナミックレンジが使用されないようにすることによって、ビデオを復号化することができる。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シグナリングが制約されていると決定し、ＯｌｓＩｎＳｃｏｐｅ内の画像の代替ライスパラメータシグナリング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化することができる。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内の画像の代替ライスパラメータシグナリング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化することができる。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内の画像の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化することができる。フラグｇｃｉ＿ｎｏ＿ｒｅｖｅｒｓｅ＿ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが１である場合、デコーダは、ＯｌｓＩｎＳｃｏｐｅ内のすべての画像のｓｐｓ＿ｒｅｖｅｒｓｅ＿ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｅｎａｂｌｅｄ＿ｆｌａｇが０に等しく、最後の有効係数の座標が現在スライスの各変換ブロックの左上隅（０，０）を基準として符号化されると決定し、復号化された最後の有効係数の座標を、現在スライスの各変換ブロックの左上隅（０，０）を基準としたものとして解釈することによって、ビデオのビットストリームの残りの部分を復号化することができる。

【０１７９】

ブロック５１０において、Ｍが５以下であると決定された場合、ブロック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、このＭビットを破棄することを含む。つまり、ビデオの復号化は、このＭビットから独立して実行される。ブロック５２０において、デコーダは、これら６つの追加符号化ツールに制約を課さずに、ビデオを復号化する。ブロック５０６において、ＧＣＩフラグがビデオに一般制約が課されていないことを示すと決定された場合（即ち、ＧＣＩフラグが０である）、プロセス５００は、ビデオを一般制約のない画像に復号化する。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、復号化を実行する。復号化済みのビデオは、表示のために出力されることができる。

【０１８０】

図６は、本発明のいくつかの実施例に係る、ビデオ復号化プロセス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、当該コンピューティング機器は、例えば、エントロピ復号化モジュール２１６、逆量子化モジュール２１８、逆変換モジュール２１９、インループ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。

【０１８１】

ブロック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）のビットストリームにアクセスすることを含む。ブロック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一般制約情報（ＧＣＩ）フラグを抽出することを含む。上記で説明したように、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指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、追加符号化ツールに課される制約を示す

ために使用されることを指定する。ｇｃｉ＿ｐｒｅｓｅｎｔ＿ｆｌａｇが０に等しい場合、ＧＣＩ構文要素が存在せず、ビデオに一般制約が課されていないことを指定する。エンコーダによっては、ＧＣＩフラグは、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抽出されることができる。

【０１８２】

ブロック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つ又は複数の一般制約が課されているか否かを決定することを含む。課されている場合（即ち、ＧＣＩフラグが１である）、ブロック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、当該フラグビットは、ビデオに対して制約される対応する追加符号化ツールを示す。

【０１８３】

ブロック６１０において、プロセス６００は、Ｍが６より大きいか否かを決定することを含む。そうである場合、ブロック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することを含み、これら６つのフラグビットは、対応するフラグを表し、これらの対応するフラグは、６つの追加符号化ツールの対応する制約を示す。これら６つのフラグは、ビデオの画像がイントラランダムアクセスポイント（ＩＲＡＰ）画像又は漸次復号リフレッシュ（ＧＤＲ）画像に制限されることを示すフラグｇｃｉ＿ａｌｌ＿ｒａｐ＿ｐｉｃｔｕｒｅｓ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇと、拡張変換精度が制約され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シグナリングが制約され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いて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化するときに、ＯｌｓＩｎＳｃｏｐｅ内の画像に制約を課すか否かを示すフラグｇｃｉ＿ｎｏ＿ｒｅｖｅｒｓｅ＿ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇと、を含む。

【０１８４】

ブロック６１４において、プロセス６００は、ビットストリームからこれら６つのフラグビットに続く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、ビデオの復号化は、このＭ－６ビットから独立して実行される。ブロック６１６において、プロセス６００は、これら６つのフラグによって示される６つの追加符号化ツールの制約に基づいて、ビデオのビットストリームの残りの部分を画像に復号化することを含む。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画像がＩＲＡＰ画像又はｐｈ＿ｒｅｃｏｖｅｒｙ＿ｐｏｃ＿ｃｎｔが０に等しいＧＤＲ画像であると決定し、前記１つ又は複数の出力レイヤセット内のＧＤＲ画像又はＩＲＡＰ画像を復号化することができる。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、ＯｌｓＩｎＳｃｏｐｅ内の画像のｓｐｓ＿ｅｘｔｅｎｄｅｄ＿ｐｒｅｃｉｓｉｏｎ＿ｆｌａｇを０に設定して、拡張ダイナミックレンジが使用されないようにすることによって、ビデオを復号化することができる。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シグナリングが制約されていると決定し、ＯｌｓＩｎＳｃｏｐｅ内の画像の代替ライスパラメータシ

グナリング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化することができる。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内の画像の代替ライスパラメータシグナリング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化することができる。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内の画像の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化することができる。フラグｇｃｉ＿ｎｏ＿ｒｅｖｅｒｓｅ＿ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｃｏｎｓｔｒａｉｎｔ＿ｆｌａｇが１である場合、デコーダは、ＯｌｓＩｎＳｃｏｐｅ内のすべての画像のｓｐｓ＿ｒｅｖｅｒｓｅ＿ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｅｎａｂｌｅｄ＿ｆｌａｇが０に等しいと決定し、最後の有効係数の座標が現在スライスの各変換ブロックの左上隅（０，０）を基準として符号化されると決定し、復号化された最後の有効係数の座標を、現在スライスの各変換ブロックの左上隅（０，０）を基準としたものとして解釈することによって、ビデオのビットストリームの残りの部分を復号化することができる。

【０１８５】

ブロック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、ブロック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。ブロック６２２において、デコーダは、抽出されたフラグビットに基づいて、６つの追加符号化ツールに制約を課すことによって、ビデオを復号化する。Ｍが０である場合、フラグビットを抽出しない。ブロック６２４において、デコーダは、フラグビットが抽出されないため、６つの追加符号化ツールに制約を課さずに、ビデオを復号化する。ブロック６０６において、ＧＣＩフラグがビデオに一般制約を課されていないことを示すと決定された場合（即ち、ＧＣＩフラグが０である）、ブロック６１８において、プロセス６００は、ビデオを一般制約のない画像に復号化することを含む。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、復号化を実行する。復号化済みのビデオは、表示のために出力されることができる。

【０１８６】

図７は、本発明のいくつかの実施例に係る、ビデオ復号化プロセス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、当該コンピューティング機器は、例えば、エントロピ復号化モジュール２１６、逆量子化モジュール２１８、逆変換モジュール２１９、インループ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。

【０１８７】

ブロック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。ブロック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一般制約情報（ＧＣＩ）フラグを抽出することを含む。上記で説明したように、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指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、追加符号化ツールに課されている制約を示すために使用されることを指定する。ｇｃｉ＿ｐｒｅｓｅｎｔ＿ｆｌａｇが０に等しい場合、ＧＣＩ構文要素が存在せず、ビデオに一般制約が課されないことを指定する。

エンコーダによっては、ＧＣＩフラグは、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抽出されることができる。

【０１８８】

ブロック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つ又は複数の一般制約が課されているか否かを決定することを含む。課されている場合（即ち、ＧＣＩフラグが１である）、ブロック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、複数のフラグビットを含み、当該複数のフラグビットは、ビデオに対して制約される対応する追加符号化ツールを示す。

【０１８９】

ブロック７１０において、プロセス７００は、Ｍが５より大きいか否かを決定することを含む。そうでない場合、ブロック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。ブロック７１４において、プロセス７００は、このＭビットから独立して、ビデオのビットストリームの残りの部分を画像に復号化することを含む。ブロック７１０において、Ｍが５以下であると決定された場合、ブロック７１８において、プロセス７００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。Ｍが６より大きい場合、ブロック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、このＭ－６ビットを破棄することを含む。つまり、ビデオの復号化は、このＭ－６ビットから独立して実行される。ブロック７２０において、デコーダは、前述のように、抽出された６つのフラグビットに基づいて、６つの追加符号化ツールに制約を課すことによって、ビデオを復号化する。

【０１９０】

ブロック７０６において、ＧＣＩフラグがビデオに一般制約が課されていないことを示すと決定された場合（即ち、ＧＣＩフラグが０である）、プロセス７００は、ビデオを一般制約のない画像に復号化することを含む。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、復号化を実行する。復号化済みのビデオは、表示のために出力されることができる。

【０１９１】

一般制約情報のシグナリング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、メモリ８１４に記憶されたコンピュータ実行可能なプログラムコード及び／又はアクセス情報を実行する。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（１つの処理機器を含む）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、命令は、プロセッサに本明細書で説明された動作を実行させる。

【０１９２】

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、任意の適切なコンピュータプログラミング言語は、例えば、Ｃ、Ｃ＋＋、Ｃ＃、Ｖｉｓｕａｌ　Ｂａｓｉｃ、Ｊａｖａ（登録商標）、Ｐｙｔｈｏｎ、Ｐｅｒｌ、ＪａｖａＳｃｒｉｐｔ、及びＡｃｔｉｏｎＳｃｒｉｐｔなどを含む。

【０１９３】

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。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、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を備えることが示されており、Ｉ／Ｏインターフェース８１８は、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。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。

【０１９４】

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、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動作の１つ又は複数の動作を実行させる。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。

【０１９５】

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。

【０１９６】

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、多数の具体的な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、これらの具体的な詳細なしにも、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。

【０１９７】

特に明記しない限り、本明細書全体では、「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、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１つ又は複数の電子コンピューティング機器）の動作又はプロセスを表し、このような動作又はプロセスは、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。

【０１９８】

本明細書で説明する１つ又は複数のシステムは、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多用途マイクロプロセッサに基づくコンピュータシステムを含み、記憶されたソフトウェアは、コンピューティングシステムをプログラミング又は構成し、これにより、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専用コンピューティング装置に変更させ、本明細書の主題の１つ又は複数の実施例を実現するようにする。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。

【０１９９】

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。上記の例で示したブロックの順序は変更でき、例えば、ブロックの順序を変更したり、結合したり、サブブロックに分割したりすることができる。いくつかのブロック又はプロセスは、並行して実行できる。

【０２００】

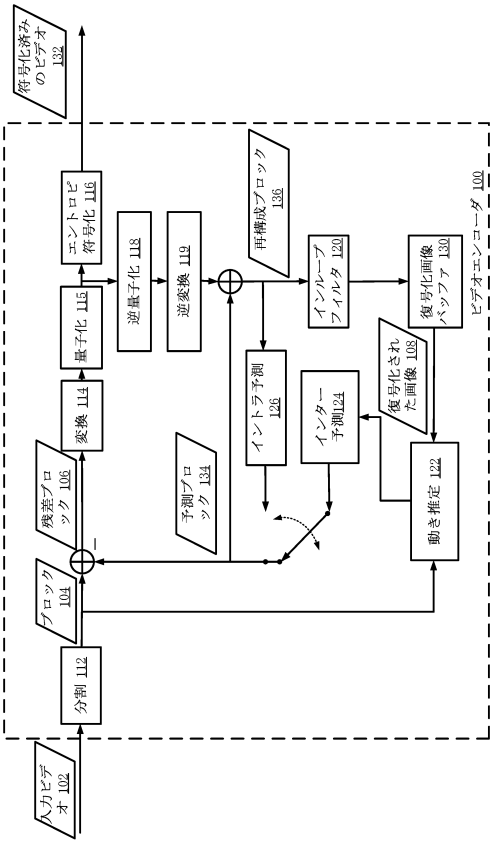
本明細書で使用される「…に適用される」又は「…に構成される」は、開放的かつ包括的な言語を意味し、追加のタスク又はステップを実行するように適用又は構成された機器を排除しない。さらに、「…に基づいて」の使用は、開放的かつ包括的であることを意図し、これは、１つ又は複数の列挙された条件又は値「に基づく」プロセス、ステップ、計算又は他の動作は、実際には、列挙されたものを超える追加の条件又は値に基づいている可能性があるためである。本明細書に含まれる見出し、リスト及び番号付けは、説明を容易にするためだけのものであり、限定することを意図しない。

【０２０１】

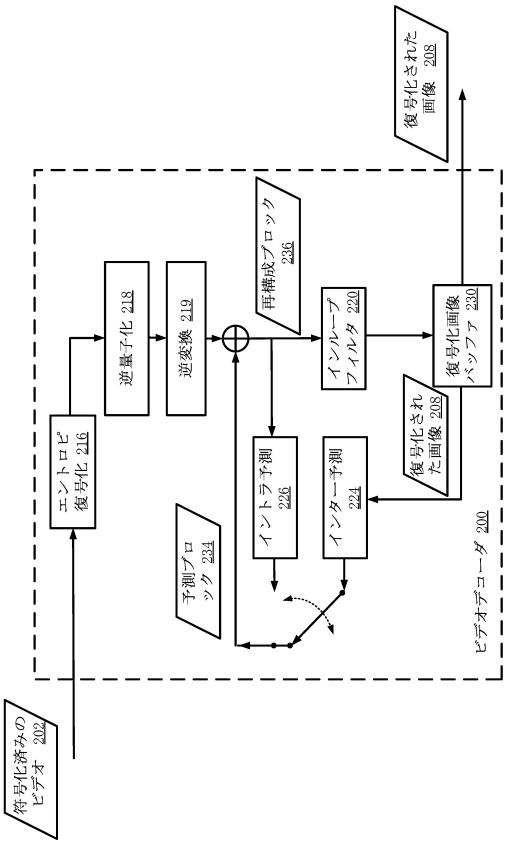
本明細書の主題については、その具体的な実施例に関して詳細に説明されているが、当業者であれば、前述の内容を理解した後、このような実施例に容易に変更、変形、及び同等のものをもたらすことができることを理解されたい。したがって、理解すべきこととして、本発明は、限定ではなく例示の目的で提示されており、当業者にとって容易になし得る、本明細書の主題に対するこのような変更、変形及び／又は追加を含むことを排除しない。

【図面】

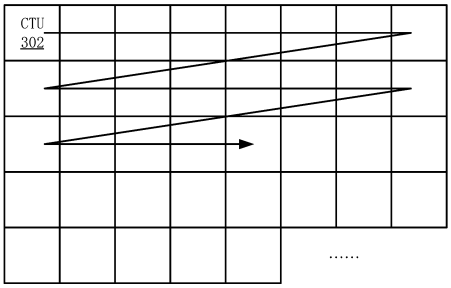
【図１】


符号化済みのビデオ 132
エントロピ符号化 116
量子化 115
逆量子化 118
逆変換 119
再構成ブロック 136
インループフィルタ 120
復号化画像バッファ 130
ビデオエンコーダ 100
変換 114
残差ブロック 106
予測ブロック 134
イントラ予測 126
インター予測 124
復号化された画像 108
動き推定 122
ブロック 104
分割 112
入力ビデオ 102

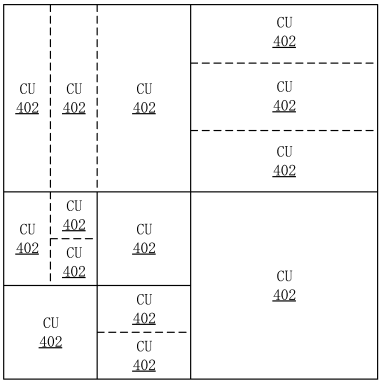
【図２】


復号化された画像 208
再構成ブロック 236
逆量子化 218
逆変換 219
インループフィルタ 220
復号化画像バッファ 230
エントロピ復号化 216
イントラ予測 226
インター予測 224
復号化された画像 208
予測ブロック 234
ビデオデコーダ 200
符号化済みのビデオ 202

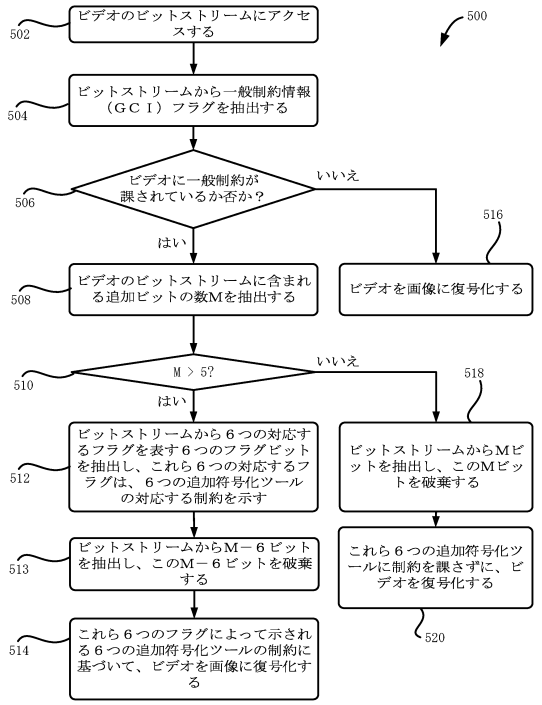
【図３】


CTU 302
……

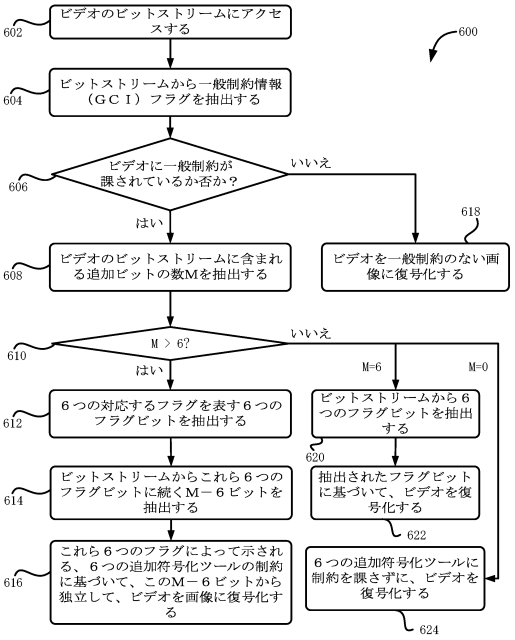
【図４】


CU 402

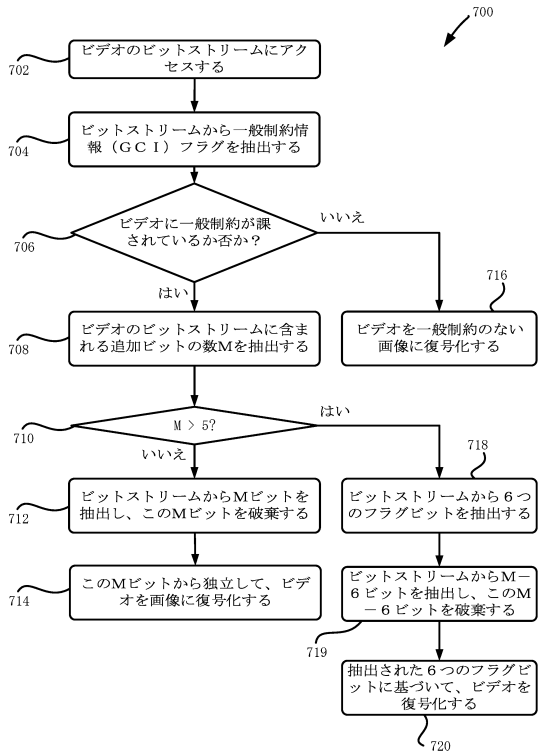
【図５】


500
502
ビデオのビットストリームにアクセスする
504
ビットストリームから一般制約情報（ＧＣＩ）フラグを抽出する
506
ビデオに一般制約が課されているか否か？
いいえ
はい
516
ビデオを画像に復号化する
508
ビデオのビットストリームに含まれる追加ビットの数Ｍを抽出する
510
Ｍ＞５?
いいえ
はい
518
512
ビットストリームから６つの対応するフラグを表す６つのフラグビットを抽出し、これら６つの対応するフラグは、６つの追加符号化ツールの対応する制約を示す
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
513
ビットストリームからＭ－６ビットを抽出し、このＭ－６ビットを破棄する
これら６つの追加符号化ツールに制約を課さずに、ビデオを復号化する
514
これら６つのフラグによって示される６つの追加符号化ツールの制約に基づいて、ビデオを画像に復号化する
520

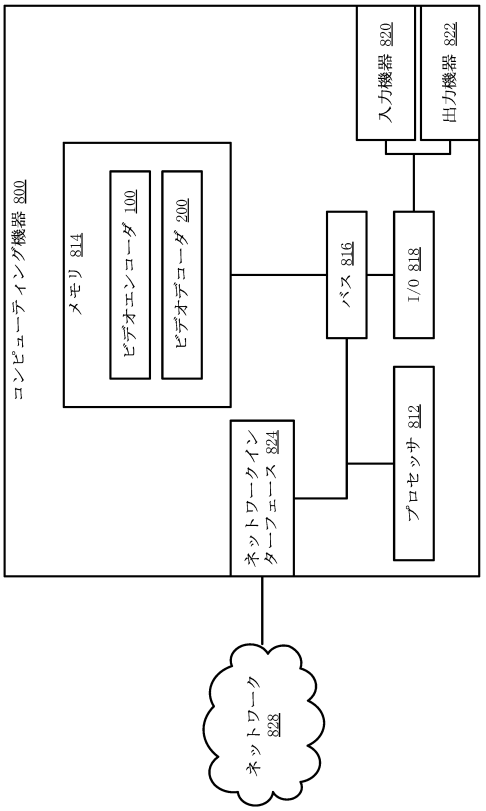
【図６】


600
602
ビデオのビットストリームにアクセスする
604
ビットストリームから一般制約情報（ＧＣＩ）フラグを抽出する
606
ビデオに一般制約が課されているか否か？
いいえ
はい
618
608
ビデオのビットストリームに含まれる追加ビットの数Ｍを抽出する
ビデオを一般制約のない画像に復号化する
610
Ｍ＞６?
いいえ
はい
Ｍ=６
Ｍ=０
612
６つの対応するフラグを表す６つのフラグビットを抽出する
ビットストリームから６つのフラグビットを抽出する
620
614
ビットストリームからこれら６つのフラグビットに続くＭ－６ビットを抽出する
抽出されたフラグビットに基づいて、ビデオを復号化する
622
616
これら６つのフラグによって示される、６つの追加符号化ツールの制約に基づいて、このＭ－６ビットから独立して、ビデオを画像に復号化する
６つの追加符号化ツールに制約を課さずに、ビデオを復号化する
624

【図７】


700
702
ビデオのビットストリームにアクセスする
704
ビットストリームから一般制約情報（ＧＣＩ）フラグを抽出する
706
ビデオに一般制約が課されているか否か？
いいえ
はい
716
708
ビデオのビットストリームに含まれる追加ビットの数Ｍを抽出する
ビデオを一般制約のない画像に復号化する
710
Ｍ＞５?
はい
いいえ
718
712
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
ビットストリームから６つのフラグビットを抽出する
714
このＭビットから独立して、ビデオを画像に復号化する
ビットストリームからＭ－６ビットを抽出し、このＭ－６ビットを破棄する
719
抽出された６つのフラグビットに基づいて、ビデオを復号化する
720

【図８】


コンピューティング機器 800
メモリ 814
ビデオエンコーダ 100
ビデオデコーダ 200
入力機器 820
出力機器 822
バス 816
I/O 818
プロセッサ 812
ネットワークインターフェース 824
ネットワーク 828

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2022/079494

A. CLASSIFICATION OF SUBJECT MATTER

IPC(8) - INV. - H04N 19/70; H04N 19/124 (2023.01)
ADD. - H04N 19/136 (2023.01)

CPC - INV. - H04N 19/70; H04N 19/124 (2023.01)

ADD. - H04N 19/136; H04N 19/159; H04N 19/119; H04N 19/172; H04N 19/174 (2023.01)

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 2021/0314587 A1 (TENCENT AMERICA LLC) 07 October 2021 (07.10.2021) entire document	1-20
A	WO 2021/160165 A1 (BEIJING BYTEDANCE NETWORK TECHNOLOGY CO. LTD. et al) 19 August 2021 (19.08.2021) 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 2017/0347128 A1 (ARRIS ENTERPRISES LLC) 30 November 2017 (30.11.2017) entire document	1-20
A	WO 2021/127723 A1 (CANON KABUSHIKI KAISHA) 01 July 2021 (01.07.2021) entire document	1-20
P, X	WO 2022/214877 A1 (INTERDIGITAL VC HOLDINGS FRANCE SAS) 13 October 2022 (13.10.2022) 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	Date of mailing of the international search report
03 January 2023	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

(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 ２４７９ 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 ２４７９ 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 ２４７９ 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　RC12