(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6779918号
(P6779918)
(45)発行日　令和2年11月4日(2020.11.4)　(24)登録日　令和2年10月16日(2020.10.16)

(51)Int.Cl.　F I
H04N 19/91 (2014.01)　H04N 19/91
H04N 19/70 (2014.01)　H04N 19/70
H04N 19/13 (2014.01)　H04N 19/13

請求項の数 14　(全 67 頁)

(21)出願番号　特願2017-561643 (P2017-561643)
(86)(22)出願日　平成28年5月27日 (2016.5.27)
(65)公表番号　特表2018-521556 (P2018-521556A)
(43)公表日　平成30年8月2日 (2018.8.2)
(86)国際出願番号　PCT/US2016/034647
(87)国際公開番号　WO2016/196287
(87)国際公開日　平成28年12月8日 (2016.12.8)
審査請求日　令和1年5月13日 (2019.5.13)
(31)優先権主張番号　62/168,503
(32)優先日　平成27年5月29日 (2015.5.29)
(33)優先権主張国・地域又は機関
米国 (US)
(31)優先権主張番号　15/166,044
(32)優先日　平成28年5月26日 (2016.5.26)
(33)優先権主張国・地域又は機関
米国 (US)

(73)特許権者　595020643
クゥアルコム・インコーポレイテッド
ＱＵＡＬＣＯＭＭ　ＩＮＣＯＲＰＯＲＡＴＥＤ
アメリカ合衆国、カリフォルニア州　９２１２１－１７１４、サン・ディエゴ、モアハウス・ドライブ　５７７５
(74)代理人　100108855
弁理士　蔵田　昌俊
(74)代理人　100109830
弁理士　福原　淑弘
(74)代理人　100158805
弁理士　井関　守三
(74)代理人　100112807
弁理士　岡田　貴志

最終頁に続く

(54)【発明の名称】高度算術コーダ

(57)【特許請求の範囲】
【請求項１】
ビデオデータのエントロピーコーディングのための方法であって、前記方法が、
コンテキスト適応型バイナリ算術コーディング（ＣＡＢＡＣ）プロセスにおいて使用される複数のコンテキストのうちの第１のコンテキストのための複数のウィンドウサイズのうちの第１のウィンドウサイズを決定することと、ここにおいて、前記第１のコンテキストのための前記第１のウィンドウサイズが、第１のシンタックス要素がコーディングされることに基づく、
前記第１のコンテキストの確率状態に基づいて、前記ビデオデータの前記第１のシンタックス要素のための値のビンをエントロピーコーディングすることと、
前記第１のコンテキストのための前記第１のウィンドウサイズと前記第１のシンタックス要素のための前記値の前記コーディングされたビンとに基づいて前記第１のコンテキストの前記確率状態を更新することと、
前記複数のコンテキストのうちの前記第１のコンテキストのための前記複数のウィンドウサイズのうちの第２のウィンドウサイズを決定することと、ここにおいて、前記第１のコンテキストのための前記第２のウィンドウサイズが、前記第１のコンテキストのための前記第１のウィンドウサイズとは異なり、前記第１のコンテキストのための前記第２のウィンドウサイズが、第２のシンタックス要素がコーディングされることに基づく、
前記第１のコンテキストの確率状態に基づいて、前記ビデオデータの第２のシンタックス要素のための値のビンをエントロピーコーディングすることと、

前記第１のコンテキストのための前記第２のウィンドウサイズと前記第２のシンタックス要素のための前記コーディングされたビンとに基づいて前記第１のコンテキストの前記確率状態を更新することと、
を備える、方法。
【請求項２】
前記第１のコンテキストの前記更新された確率状態に基づいて、前記第１のコンテキストに関連する別のビンをエントロピーコーディングすることをさらに備える、請求項１に記載の方法。
【請求項３】
前記第１のコンテキストのための前記第１のおよび第２のウィンドウサイズが、前記コーディングされたビンを含むビットストリーム中でシグナリングされない、請求項１に記載の方法。
【請求項４】
前記複数のウィンドウサイズが、ウィンドウサイズのあらかじめ定義されたセットを備える、請求項１に記載の方法。
【請求項５】
エントロピーコーディングすることがエントロピー符号化することを備え、ここにおいて、前記ウィンドウサイズを決定することが、
ウィンドウサイズの前記あらかじめ定義されたセットのそれぞれのウィンドウサイズについて、前記シンタックス要素のためのビン値を含む特定のビンストリングをエントロピー符号化するために使用されるビットのそれぞれの量を決定することと、
前記特定のビンストリングをエントロピー符号化するために、ビットの最も小さい量に対応する、ウィンドウサイズの前記あらかじめ定義されたセットのうちの前記ウィンドウサイズを前記第１のコンテキストのための前記ウィンドウサイズとして選択することと
を備える、請求項４に記載の方法。
【請求項６】
デフォルトウィンドウサイズが前記複数のコンテキストのために使用されるかどうかを示す第３のシンタックス要素をコーディングすることをさらに備える、請求項１に記載の方法。
【請求項７】
前記デフォルトウィンドウサイズが前記複数のコンテキストのために使用されないことを前記第３のシンタックス要素が示すことに基づいて、前記第１のコンテキストのための前記ウィンドウサイズを示す第４のシンタックス要素をコーディングすることをさらに備える、請求項６に記載の方法。
【請求項８】
前記第１のコンテキストのための前記ウィンドウサイズを示すために、前記第４のシンタックス要素が、前記第１のコンテキストのための前記ウィンドウサイズと前記デフォルトウィンドウサイズとの間の差分を示す、請求項７に記載の方法。
【請求項９】
前記第３のシンタックス要素をコーディングすることが、前記第３のシンタックス要素を含む現在スライスのスライスヘッダをコーディングすることを備え、ここにおいて、前記第３のシンタックス要素は、前記現在スライスのビンをエントロピーコーディングするとき、前記デフォルトウィンドウサイズが前記複数のコンテキストのために使用されるかどうかを示す、請求項６に記載の方法。
【請求項１０】
現在スライスのスライスヘッダ中で、前記複数のコンテキストのためのウィンドウサイズが、前にコーディングされたスライスから継承されるかどうかを示すシンタックス要素をコーディングすることをさらに備える、請求項１に記載の方法。
【請求項１１】
エントロピーコーディングすることがエントロピー復号することを備え、前記方法が、

コード化ビデオビットストリームから、前記第１のコンテキストのための前記ウィンドウサイズを示す１つまたは複数のシンタックス要素を復号することをさらに備える、請求項１に記載の方法。

【請求項１２】

ビデオデータのエントロピーコーディングのための装置であって、前記装置が、

コンテキスト適応型バイナリ算術コーディング（ＣＡＢＡＣ）プロセスにおいて使用される複数のコンテキストのうちの第１のコンテキストのための複数のウィンドウサイズのうちの第１のウィンドウサイズを決定するための手段と、ここにおいて、前記第１のコンテキストのための前記第１のウィンドウサイズが、第１のシンタックス要素がコーディングされることに基づく、

前記第１のコンテキストの確率状態に基づいて、前記ビデオデータの前記第１のシンタックス要素のための値のビンをエントロピーコーディングするための手段と、

前記第１のコンテキストのための前記第１のウィンドウサイズと前記第１のシンタックス要素のための前記値の前記コーディングされたビンとに基づいて前記第１のコンテキストの前記確率状態を更新するための手段と、

前記複数のコンテキストのうちの前記第１のコンテキストのための前記複数のウィンドウサイズのうちの第２のウィンドウサイズを決定するための手段と、ここにおいて、前記第１のコンテキストのための前記第２のウィンドウサイズが、前記第１のコンテキストのための前記第１のウィンドウサイズとは異なる、

前記第１のコンテキストの確率状態に基づいて、前記ビデオデータの第２のシンタックス要素のための値のビンをエントロピーコーディングするための手段と、

前記第１のコンテキストのための前記第２のウィンドウサイズと前記第２のシンタックス要素のための前記コーディングされたビンとに基づいて前記第１のコンテキストの前記確率状態を更新するための手段と、

を備える、装置。

【請求項１３】

実行されたとき、ビデオコーディングデバイスの１つまたは複数のプロセッサに、請求項１～１０のうちのいずれかに記載の方法を実行させる命令を記憶するコンピュータ可読記憶媒体。

【請求項１４】

実行されたとき、ビデオコーディングデバイスの１つまたは複数のプロセッサに、請求項１１に記載の方法を実行させる命令を記憶するコンピュータ可読記憶媒体。

【発明の詳細な説明】

【技術分野】

【０００１】

[0001] 本出願は、その内容全体が参照により本明細書に組み込まれる、２０１５年５月２９日に出願された米国仮出願第６２／１６８，５０３号の利益を主張する。

【０００２】

[0002] 本開示は、ビデオコーディング（video coding）に関し、より詳細には、ビデオデータ（video data）のバイナリ算術コーディング（binary arithmetic coding）のための技法に関する。

【背景技術】

【０００３】

[0003] デジタルビデオ機能は、デジタルテレビジョン、デジタルダイレクトブロードキャストシステム、ワイヤレスブロードキャストシステム、携帯情報端末（ＰＤＡ）、ラップトップまたはデスクトップコンピュータ、デジタルカメラ、デジタル記録デバイス、デジタルメディアプレーヤ、ビデオゲームデバイス、ビデオゲームコンソール、セルラーまたは衛星無線電話、ビデオ遠隔会議デバイスなどを含む、広範囲にわたるデバイスに組み込まれ得る。デジタルビデオデバイスは、デジタルビデオ情報をより効率的に送信、受信および記憶する（store）ための、ＭＰＥＧ－２、ＭＰＥＧ－４、ＩＴＵ－Ｔ　Ｈ．２

６３、ＩＴＵ－Ｔ Ｈ．２６４／ＭＰＥＧ－４，Ｐａｒｔ１０，アドバンストビデオコーディング（ＡＶＣ：Advanced Video Coding）によって定義された規格、高効率ビデオコーディング（ＨＥＶＣ：High Efficiency Video Coding）規格、およびそのような規格の拡張に記載されているビデオ圧縮技法など、ビデオ圧縮技法を実装する。

【０００４】

[0004] ビデオ圧縮技法は、ビデオシーケンスに固有の冗長性を低減または除去するための空間予測および／または時間予測を含む。ブロックベースのビデオコーディングの場合、ビデオフレームまたはスライスはブロックに区分され得る。各ブロックはさらに区分され得る。イントラコード化（Ｉ）フレームまたはスライス中のブロックは、同じフレームまたはスライス中の隣接ブロック中の参照サンプルに対する空間予測を使用して符号化される。インターコード化（ＰまたはＢ）フレームまたはスライス中のブロックは、同じフレームまたはスライス中の隣接ブロック中の参照サンプルに対する空間予測、あるいは他の参照フレーム中の参照サンプルに対する時間予測を使用し得る。空間予測または時間予測は、コーディングされるべきブロックのための予測ブロックを生じる。残差データ（Residual data）は、コーディングされるべき元のブロックと予測ブロックとの間のピクセル差分を表す。

【０００５】

[0005] インターコード化ブロック（inter-coded block）は、予測ブロックを形成する参照サンプルのブロックを指す動きベクトル（motion vector）と、コード化ブロックと予測ブロックとの間の差分（difference）を示す残差データとに従って符号化される。イントラコード化ブロックは、イントラコーディングモードと残差データとに従って符号化される。さらなる圧縮のために、残差データは、ピクセル領域から変換領域に変換され、残差変換係数（residual transform coefficient）が生じ得、その残差変換係数は、次いで量子化され得る。最初は２次元アレイに構成される量子化された変換係数は、エントロピーコーディング（entropy coding）のための変換係数の１次元ベクトルを生成するために、特定の順序で走査され得る。

【０００６】

[0006] 様々なエントロピーコーディングプロセスが、残差変換係数、動きベクトル情報、シンタックス要素（syntax element）、および他の関連する情報をコーディングするために利用得る。様々なエントロピーコーディングおよび他のデータ圧縮プロセスの例としては、コンテキスト適応型可変長コーディング（ＣＡＶＬＣ：context-adaptive variable length coding）、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）、確率間隔区分エントロピーコーディング（ＰＩＰＥ：probability interval partitioning entropy coding）、ゴロムコーディング（Golomb coding）、ゴロム－ライスコーディング（Golomb-Rice coding）、および指数ゴロムコーディング（exponential Golomb coding）がある。

【発明の概要】

【０００７】

[0007] 概して、本開示は、ビデオコーディングを実行するための技法について説明する。より詳細には、本開示は、異なるウィンドウサイズ（window size）でコンテキストベースエントロピーコーディング（context-based entropy coding）を実行するための例示的な技法について説明する。いくつかの例では、本開示で説明される技法は、異なるウィンドウサイズでのＣＡＢＡＣの実行を可能にし得る。他の例では、本開示で説明される技法は、コンテキストベース可変長コーディング（context-based variable length coding）など、シンボル（symbol）をコーディングするためにコンテキスト（context）を使用する他のエントロピーコーダ（entropy coder）に適用され得る。

【０００８】

[0008] 一例では、ビデオデータのエントロピーコーディングのための方法は、シンタックス要素のための値（value）をエントロピーコーディングするために、コンテキスト適応型エントロピーコーディングプロセス（context-adaptive entropy coding process

）（たとえば、ＣＡＢＡＣまたはＣＡＶＬＣプロセス）において使用される複数のコンテキストのうちの１つのコンテキストのための複数のウィンドウサイズのうちの１つのウィンドウサイズを決定する（determine）ことを含む。この例では、本方法はまた、シンタックス要素のための値のビン（bin）をエントロピーコーディングすることと、ウィンドウサイズとコーディングされたビン（coded bin）とに基づいてコンテキストの確率状態（probability state）を更新する（update）こととを含む。この例では、本方法はまた、コンテキストの更新された確率状態に基づいて、同じコンテキストをもつ次のビンをエントロピーコーディングすることを含む。

【０００９】

[0009] 別の例では、ビデオデータのエントロピーコーディングのための装置（apparatus）は、１つまたは複数のプロセッサ（processor）と、ビデオデータのシンタックス要素のための値をエントロピーコーディングするために、コンテキスト適応型エントロピーコーディングプロセスにおいて使用される複数のコンテキストを記憶するように構成されたメモリ（memory）とを含む。この例では、１つまたは複数のプロセッサは、複数のコンテキストのうちのコンテキストのためのウィンドウサイズを決定するように構成される。この例では、１つまたは複数のプロセッサは、コンテキストの確率状態に基づいて、シンタックス要素のための値のビンをエントロピーコーディングすることと、ウィンドウサイズとコーディングされたビンとに基づいてコンテキストの確率状態を更新することと、コンテキストの更新された確率状態に基づいて、コンテキストの更新された確率状態に基づいて、同じコンテキストをもつ次のビンをコーディングすることとを行うようにさらに構成される。

【００１０】

[0010] 別の例では、ビデオデータのエントロピーコーディングのための装置は、ビデオデータのシンタックス要素のための値をエントロピーコーディングするために、コンテキスト適応型コーディングプロセス（context-adaptive coding process）において使用される複数のコンテキストのうちのコンテキストのための複数のウィンドウサイズのうちのウィンドウサイズを決定するための手段と、コンテキストの確率状態に基づいて、シンタックス要素のための値のビンをエントロピーコーディングするための手段と、ウィンドウサイズとコーディングされたビンとに基づいてコンテキストモデル（context model）の確率状態を更新するための手段とを含む。この例では、本装置はまた、コンテキストモデルの更新された確率状態に基づいて、同じコンテキストをもつ次のビンをエントロピーコーディングするための手段を含む。

【００１１】

[0011] 別の例では、コンピュータ可読記憶媒体（computer-readable storage medium）は、実行されたとき、ビデオコーディングデバイスの１つまたは複数のプロセッサに、ビデオデータのシンタックス要素のための値をエントロピーコーディングするために、コンテキスト適応型コーディングプロセスにおいて使用される複数のコンテキストのうちのコンテキストのための複数のウィンドウサイズのうちのウィンドウサイズを決定することと、コンテキストの確率状態に基づいて、シンタックス要素のための値のビンをエントロピーコーディングすることと、ウィンドウサイズとコーディングされたビンとに基づいてコンテキストの確率状態を更新することと、コンテキストモデルの更新された確率状態に基づいて、同じコンテキストをもつ次のビンをエントロピーコーディングすることとを行わせる命令（instruction）を記憶する。

【００１２】

[0012] 別の例では、コンピュータ可読記憶媒体は、ビデオ復号デバイス（video decoding device）によって処理されたとき、ビデオ復号デバイスの１つまたは複数のプロセッサに、シンタックス要素のための値をエントロピーコーディングするために、コンテキスト適応型コーディングプロセスにおいて使用される複数のコンテキストのうちのコンテキストのための複数のウィンドウサイズのうちのウィンドウサイズを決定することと、コンテキストの確率状態に基づいて、シンタックス要素のための値のビンをエントロピーコ

ーディングすることと、ウィンドウサイズとコーディングされたビンとに基づいてコンテキストの確率状態を更新することと、コンテキストモデルの更新された確率状態に基づいて、同じコンテキストをもつ次のビンをエントロピーコーディングすることとを行わせるビデオデータを記憶する。

【００１３】

[0013] 本開示の１つまたは複数の態様の詳細が添付の図面および以下の説明に記載されている。本開示で説明される技法の他の特徴、目的、および利点は、その説明および図面、ならびに特許請求の範囲から明らかになろう。

【図面の簡単な説明】

【００１４】

【図１】[0014] 例示的なビデオ符号化および復号システム（video encoding and decoding system）を示すブロック図。

【図２Ａ】[0015] バイナリ算術コーディングにおける範囲更新プロセス（range update process）を示す概念図。

【図２Ｂ】バイナリ算術コーディングにおける範囲更新プロセスを示す概念図。

【図３】[0016] バイナリ算術コーディングにおける出力プロセスを示す概念図。

【図４】[0017] 例示的なビデオエンコーダ（video encoder）を示すブロック図。

【図５】[0018] ビデオエンコーダ中のコンテキスト適応型バイナリ算術コーダ（context adaptive binary arithmetic coder）を示すブロック図。

【図６】[0019] 例示的なビデオデコーダ（video decoder）を示すブロック図。

【図７】[0020] ビデオデコーダ中のコンテキスト適応型バイナリ算術コーダを示すブロック図。

【図８】[0021] 通常コーディングモード（regular coding mode）を使用する所与のビン値のためのバイナリ算術符号化プロセス（binary arithmetic encoding process）を示す図。

【図９】[0022] 残差４分木（residual quadtree）に基づく例示的な変換方式を示す概念図。

【図１０】[0023] 係数グループ（coefficient group）に基づく例示的な係数走査（coefficient scan）を示す概念図。

【図１１】[0024] 本開示の１つまたは複数の技法による、異なるウィンドウサイズでコンテキストベースエントロピー符号化（context-based entropy encoding）を実行するための例示的なプロセスを示すフローチャート。

【図１２】[0025] 本開示の１つまたは複数の技法による、異なるウィンドウサイズでコンテキストベースエントロピー復号（context-based entropy decoding）を実行するための例示的なプロセスを示すフローチャート。

【発明を実施するための形態】

【００１５】

[0026] 本開示の技法は、概して、ブロックベースハイブリッドビデオコーディング（block-based hybrid video coding）におけるエントロピーコーディングモジュールに関する。これらの技法は、ＨＥＶＣ（高効率ビデオコーディング（High Efficiency Video Coding））など、任意の既存のビデオコーデック（video codec）に適用され得、あるいはこれらの技法は、任意の将来のビデオコーディング規格または他のプロプライエタリ（proprietary）もしくは非プロプライエタリコーディング技法における効率的なコーディングツールであり得る。例および説明のために、本開示の技法は、概して、ＨＥＶＣ（またはＩＴＵ－Ｔ　Ｈ．２６５）および／またはＩＴＵ－Ｔ　Ｈ．２６４に関して説明される。さらに、例および説明のために、本開示の技法は、概してＣＡＢＡＣコーダに関して説明されるが、本開示の技法は、コンテキスト適応型可変長コーダ（context-adaptive variable-length coder）など、他のコンテキストベースエントロピーコーダ（context-based entropy coder）に適用可能であり得ることを理解されたい。

【００１６】

[0027] 図１は、可変ウィンドウサイズ（variable window size）でＣＡＢＡＣ設計に従ってデータをコーディングするための技法を利用し得る例示的なビデオ符号化および復号システム（video encoding and decoding system）１０を示すブロック図である。図１に示されているように、システム１０は、宛先デバイス１４によって後で復号されるべき符号化ビデオデータ（encoded video data）を与えるソースデバイス１２を含む。特に、ソースデバイス１２は、コンピュータ可読媒体１６を介してビデオデータを宛先デバイス１４に与える。ソースデバイス１２および宛先デバイス１４は、デスクトップコンピュータ、ノートブック（すなわち、ラップトップ）コンピュータ、タブレットコンピュータ、セットトップボックス、いわゆる「スマート」フォンなどの電話ハンドセット、いわゆる「スマート」パッド、テレビジョン、カメラ、ディスプレイデバイス、デジタルメディアプレーヤ、ビデオゲームコンソール、ビデオストリーミングデバイスなどを含む、広範囲にわたるデバイスのいずれかを備え得る。いくつかの場合には、ソースデバイス１２および宛先デバイス１４は、ワイヤレス通信のために装備され得る。

【００１７】

[0028] 宛先デバイス１４は、コンピュータ可読媒体１６を介して復号されるべき符号化ビデオデータを受信し得る。コンピュータ可読媒体１６は、ソースデバイス１２から宛先デバイス１４に符号化ビデオデータを移動させることが可能な任意のタイプ（type）の媒体またはデバイスを備え得る。一例では、コンピュータ可読媒体１６は、ソースデバイス１２が、符号化ビデオデータを宛先デバイス１４にリアルタイムで直接送信することを可能にするための通信媒体を備え得る。符号化ビデオデータは、ワイヤレス通信プロトコルなどの通信規格に従って変調され、宛先デバイス１４に送信され得る。通信媒体は、無線周波数（ＲＦ）スペクトルまたは１つまたは複数の物理伝送線路など、任意のワイヤレスまたはワイヤード通信媒体を備え得る。通信媒体は、ローカルエリアネットワーク、ワイドエリアネットワーク、またはインターネットなどのグローバルネットワークなど、パケットベースネットワークの一部を形成し得る。通信媒体は、ソースデバイス１２から宛先デバイス１４への通信を可能にするために有用であり得るルータ、スイッチ、基地局、または任意の他の機器を含み得る。

【００１８】

[0029] いくつかの例では、符号化データは、出力インターフェース２２からストレージデバイスに出力され得る。同様に、符号化データは、入力インターフェースによってストレージデバイスからアクセスされ得る。ストレージデバイスは、ハードドライブ、Ｂｌｕ－ｒａｙ（登録商標）ディスク、ＤＶＤ、ＣＤ－ＲＯＭ、フラッシュメモリ、揮発性または不揮発性メモリ、あるいは符号化ビデオデータを記憶するための任意の他の好適なデジタル記憶媒体など、様々な分散されたまたはローカルにアクセスされるデータ記憶媒体のいずれかを含み得る。さらなる一例では、ストレージデバイスは、ソースデバイス１２によって生成された符号化ビデオを記憶し得るファイルサーバまたは別の中間ストレージデバイスに対応し得る。宛先デバイス１４は、ストリーミングまたはダウンロードを介して、ストレージデバイスから記憶されたビデオデータにアクセスし得る。ファイルサーバは、符号化ビデオデータを記憶することと、その符号化ビデオデータを宛先デバイス１４に送信することとが可能な任意のタイプのサーバであり得る。例示的なファイルサーバとしては、（たとえば、ウェブサイトのための）ウェブサーバ、ＦＴＰサーバ、ネットワーク接続ストレージ（ＮＡＳ：network attached storage）デバイス、またはローカルディスクドライブがある。宛先デバイス１４は、インターネット接続を含む、任意の標準のデータ接続を通して符号化ビデオデータにアクセスし得る。これは、ファイルサーバに記憶された符号化ビデオデータにアクセスするのに好適であるワイヤレスチャネル（たとえば、Ｗｉ－Ｆｉ（登録商標）接続）、ワイヤード接続（たとえば、ＤＳＬ、ケーブルモデムなど）、またはその両方の組合せを含み得る。ストレージデバイスからの符号化ビデオデータの送信は、ストリーミング送信、ダウンロード送信、またはそれらの組合せであり得る。

【００１９】

[0030] 本開示の技法は、必ずしもワイヤレス適用例または設定に限定されるとは限らない。本技法は、オーバージエアテレビジョン放送、ケーブルテレビジョン送信、衛星テレビジョン送信、動的適応ストリーミングオーバーＨＴＴＰ（ＤＡＳＨ：dynamic adaptive streaming over HTTP）などのインターネットストリーミングビデオ送信、データ記憶媒体上に符号化されたデジタルビデオ、データ記憶媒体に記憶されたデジタルビデオの復号、または他の適用例など、様々なマルチメディア適用例のいずれかをサポートするビデオコーディングに適用され得る。いくつかの例では、システム１０は、ビデオストリーミング、ビデオ再生、ビデオブロードキャスティング、および／またはビデオテレフォニーなどの適用例をサポートするために、一方向または双方向のビデオ送信をサポートするように構成され得る。

【００２０】

[0031] 図１の例では、ソースデバイス１２は、ビデオソース１８と、ビデオエンコーダ２０と、出力インターフェース２２とを含む。宛先デバイス１４は、入力インターフェース２８と、ビデオデコーダ３０と、ディスプレイデバイス３１とを含む。本開示によれば、ソースデバイス１２のビデオエンコーダ２０は、拡張ＣＡＢＡＣ設計に従ってデータをコーディングするための技法を適用するように構成され得る。他の例では、ソースデバイスおよび宛先デバイスは他の構成要素または構成を含み得る。たとえば、ソースデバイス１２は、外部カメラなど、外部ビデオソース１８からビデオデータを受信し得る。同様に、宛先デバイス１４は、内蔵ディスプレイデバイスを含むのではなく、外部ディスプレイデバイスとインターフェースし得る。

【００２１】

[0032] 図１の図示のシステム１０は一例にすぎない。拡張ＣＡＢＡＣ設計に従ってデータをコーディングするための技法は、任意のデジタルビデオ符号化および／または復号デバイスによって実行され得る。概して、本開示の技法はビデオ符号化デバイスによって実行されるが、本技法は、一般に「コーデック（CODEC）」と呼ばれるビデオエンコーダ／デコーダによっても実行され得る。その上、本開示の技法はビデオプリプロセッサによっても実行され得る。ソースデバイス１２および宛先デバイス１４は、ソースデバイス１２が宛先デバイス１４に送信するためのコード化ビデオデータを生成するような、コーディングデバイスの例にすぎない。いくつかの例では、デバイス１２、１４は、デバイス１２、１４の各々がビデオ符号化構成要素とビデオ復号構成要素（video encoding and decoding components）とを含むように、実質的に対称的に動作し得る。したがって、システム１０は、たとえば、ビデオストリーミング、ビデオ再生、ビデオブロードキャスティング、またはビデオテレフォニーのための、ビデオデバイス１２とビデオデバイス１４との間の一方向または双方向のビデオ送信をサポートし得る。

【００２２】

[0033] ソースデバイス１２のビデオソース１８は、ビデオカメラなどのビデオキャプチャデバイス、以前にキャプチャ（capture）されたビデオを含んでいるビデオアーカイブ、および／またはビデオコンテンツプロバイダからビデオを受信するためのビデオフィードインターフェースを含み得る。さらなる代替として、ビデオソース１８は、ソースビデオとしてのコンピュータグラフィックスベースデータ、またはライブビデオとアーカイブビデオとコンピュータ生成ビデオとの組合せを生成し得る。いくつかの場合には、ビデオソース１８がビデオカメラである場合、ソースデバイス１２および宛先デバイス１４は、いわゆるカメラフォンまたはビデオフォンを形成し得る。ただし、上述のように、本開示で説明される技法は、概してビデオコーディングに適用可能であり得、ワイヤレスおよび／またはワイヤード適用例に適用され得る。各場合において、キャプチャされたビデオ、前にキャプチャされたビデオ、またはコンピュータ生成ビデオは、ビデオエンコーダ２０によって符号化され得る。符号化ビデオ情報は、次いで、出力インターフェース２２によってコンピュータ可読媒体１６上に出力され得る。

【００２３】

[0034] コンピュータ可読媒体（Computer-readable medium）１６は、ワイヤレスブロ

ードキャストまたはワイヤードネットワーク送信などの一時媒体、あるいはハードディスク、フラッシュドライブ、コンパクトディスク、デジタルビデオディスク、Ｂｌｕ－ｒａｙディスク、または他のコンピュータ可読媒体などの非一時的（non-transient）記憶媒体（すなわち、非一時的（non-transitory）記憶媒体）を含み得る。いくつかの例では、ネットワークサーバ（図示せず）は、たとえば、ネットワーク送信を介して、ソースデバイス１２から符号化ビデオデータを受信し、その符号化ビデオデータを宛先デバイス１４に与え得る。同様に、ディスクスタンピング設備など、媒体製造設備のコンピューティングデバイスは、ソースデバイス１２から符号化ビデオデータを受信し、その符号化ビデオデータを含んでいるディスクを生成し得る。ビデオ復号デバイスによって処理されるとき、ディスク上の符号化ビデオデータは、ビデオ復号デバイスに、本明細書で開示される様々な例に従ってビデオデータを復号させ得る。したがって、コンピュータ可読媒体１６は、様々な例において、様々な形態の１つまたは複数のコンピュータ可読媒体を含むことが理解されよう。

【００２４】

[0035] 宛先デバイス１４の入力インターフェース２８は、コンピュータ可読媒体１６から情報を受信する。コンピュータ可読媒体１６の情報は、ビデオエンコーダ２０によって定義され、またビデオデコーダ３０によって使用される、ブロックおよび他のコード化ユニット、たとえば、ＧＯＰの特性および／または処理を記述するシンタックス要素を含む、シンタックス情報を含み得る。ディスプレイデバイス３２は、復号ビデオデータ（decoded video data）をユーザに対して表示し、陰極線管（ＣＲＴ）、液晶ディスプレイ（ＬＣＤ）、プラズマディスプレイ、有機発光ダイオード（ＯＬＥＤ）ディスプレイ、または別のタイプのディスプレイデバイスなど、様々なディスプレイデバイスのいずれかを備え得る。

【００２５】

[0036] ビデオエンコーダ２０およびビデオデコーダ３０は、ＩＴＵ－Ｔ Ｈ．２６５とも呼ばれる、高効率ビデオコーディング（ＨＥＶＣ）規格など、ビデオコーディング規格に従って動作し得る。代替的に、ビデオエンコーダ２０およびビデオデコーダ３０は、代替的にＭＰＥＧ－４，Ｐａｒｔ１０，アドバンストビデオコーディング（ＡＶＣ）と呼ばれるＩＴＵ－Ｔ Ｈ．２６４規格など、他のプロプライエタリ規格または業界規格（proprietary or industry standards）、あるいはそのような規格の拡張に従って動作し得る。ただし、本開示の技法は、いかなる特定のコーディング規格にも限定されない。ビデオコーディング規格の他の例としては、ＭＰＥＧ－２およびＩＴＵ－Ｔ Ｈ．２６３がある。図１には示されていないが、いくつかの態様では、ビデオエンコーダ２０およびビデオデコーダ３０は、それぞれ、オーディオエンコーダおよびデコーダと統合され得、共通のデータストリームまたは別個のデータストリーム中のオーディオとビデオの両方の符号化を処理するために、適切なＭＵＸ－ＤＥＭＵＸユニット、または他のハードウェアおよびソフトウェアを含み得る。適用可能な場合、ＭＵＸ－ＤＥＭＵＸユニットは、ＩＴＵ Ｈ．２２３マルチプレクサプロトコル、またはユーザデータグラムプロトコル（ＵＤＰ：user datagram protocol）などの他のプロトコルに準拠し得る。

【００２６】

[0037] ビデオエンコーダ２０およびビデオデコーダ３０はそれぞれ、１つまたは複数のマイクロプロセッサ、デジタル信号プロセッサ（ＤＳＰ）、特定用途向け集積回路（ＡＳＩＣ）、フィールドプログラマブルゲートアレイ（ＦＰＧＡ）、ディスクリート論理、ソフトウェア、ハードウェア、ファームウェアなど、様々な好適なエンコーダ回路のいずれか、あるいはそれらの任意の組合せとして実装され得る。本技法が部分的にソフトウェアで実装されるとき、デバイスは、ソフトウェアのための命令を好適な非一時的コンピュータ可読媒体に記憶し、本開示の技法を実行するために１つまたは複数のプロセッサを使用してハードウェアでその命令を実行し得る。ビデオエンコーダ２０およびビデオデコーダ３０の各々は１つまたは複数のエンコーダまたはデコーダ中に含まれ得、そのいずれも、それぞれのデバイスにおいて複合エンコーダ／デコーダ（コーデック）の一部として統

合され得る。

【００２７】

[0038] 概して、ＨＥＶＣによれば、ビデオフレームまたはピクチャは、ルーマサンプルとクロマサンプルの両方を含む一連のツリーブロックまたは最大コーディングユニット（ＬＣＵ：largest coding unit）に分割され得る。ビットストリーム（bitstream）内のシンタックスデータが、ピクセルの数に関して最大コーディングユニットであるＬＣＵのサイズを定義し得る。スライスは、コーディング順序でいくつかの連続するツリーブロックを含む。ビデオフレームまたはピクチャは、１つまたは複数のスライスに区分され得る。各ツリーブロックは、４分木に従ってコーディングユニット（ＣＵ：coding unit）にスプリットされ得る。概して、４分木データ構造はＣＵごとに１つのノードを含み、ルートノードはツリーブロックに対応する。ＣＵが４つのサブＣＵにスプリットされた場合、ＣＵに対応するノードは４つのリーフノード（leaf node）を含み、リーフノードの各々はサブＣＵのうちの１つに対応する。

【００２８】

[0039] ４分木データ構造の各ノードは、対応するＣＵのためのシンタックスデータを与え得る。たとえば、４分木中のノードは、そのノードに対応するＣＵがサブＣＵにスプリットされるかどうかを示すスプリットフラグ（split flag）を含み得る。ＣＵのためのシンタックス要素は、再帰的に定義され得、ＣＵがサブＣＵにスプリットされるかどうかに依存し得る。ＣＵがさらにスプリットされない場合、そのＣＵはリーフＣＵと呼ばれる。本開示では、元のリーフＣＵの明示的スプリッティングが存在しない場合でも、リーフＣＵの４つのサブＣＵはリーフＣＵとも呼ばれる。たとえば、１６×１６サイズのＣＵがさらにスプリットされない場合、その１６×１６ＣＵが決してスプリットされなくても、４つの８×８サブＣＵはリーフＣＵとも呼ばれる。

【００２９】

[0040] ＣＵは、ＣＵがサイズ差異を有しないことを除いて、Ｈ．２６４規格のマクロブロックと同様の目的を有する。たとえば、ツリーブロックは、（サブＣＵとも呼ばれる）４つの子ノードにスプリットされ得、各子ノードは、今度は親ノードとなり、別の４つの子ノードにスプリットされ得る。４分木のリーフノードと呼ばれる、最後のスプリットされていない子ノードは、リーフＣＵとも呼ばれるコーディングノードを備える。コード化ビットストリームに関連するシンタックスデータは、最大ＣＵ深度（maximum CU depth）と呼ばれる、ツリーブロックがスプリットされ得る最大回数を定義し得、また、コーディングノードの最小サイズを定義し得る。それに応じて、ビットストリームは最小コーディングユニット（ＳＣＵ：smallest coding unit）をも定義し得る。本開示は、ＨＥＶＣのコンテキストにおけるＣＵ、予測ユニット（ＰＵ：prediction unit）、または変換ユニット（ＴＵ：transform unit）、あるいは他の規格のコンテキストにおける同様のデータ構造（たとえば、Ｈ．２６４／ＡＶＣにおけるマクロブロックおよびそれのサブブロック）のいずれかを指すために「ブロック（block）」という用語を使用する。

【００３０】

[0041] ＣＵは、コーディングノードと、コーディングノードに関連する予測ユニット（ＰＵ）および変換ユニット（ＴＵ）とを含む。ＣＵのサイズは、コーディングノードのサイズに対応し、概して形状が正方形である。ＣＵのサイズは、８×８ピクセルから最大サイズ、たとえば、６４×６４以上のピクセルをもつツリーブロックのサイズまでに及び得る。各ＣＵは、１つまたは複数のＰＵと、１つまたは複数のＴＵとを含んでいることがある。ＣＵに関連するシンタックスデータは、たとえば、１つまたは複数のＰＵへのＣＵの区分を記述し得る。区分モードは、ＣＵが、スキップモード符号化またはダイレクトモード符号化されるか、イントラ予測モード符号化されるか、あるいはインター予測モード符号化されるかの間で異なり得る。ＰＵは、形状が非正方形になるように区分され得る。ＣＵに関連するシンタックスデータは、たとえば、４分木に従う１つまたは複数のＴＵへのＣＵの区分をも記述し得る。ＴＵは、形状が正方形または非正方形（たとえば、矩形）であり得る。

【００３１】
[0042] ＨＥＶＣ規格は、ＣＵごとに異なり得るＴＵに従う変換を可能にする。ＴＵは、一般に、区分されたＬＣＵについて定義された所与のＣＵ内のＰＵのサイズに基づいてサイズ決定されるが、これは常にそうであるとは限らない。ＴＵは、一般に、ＰＵと同じサイズであるかまたはＰＵよりも小さい。いくつかの例では、ＣＵに対応する残差サンプルは、「残差４分木」（ＲＱＴ：residual quad tree）として知られる４分木構造を使用して、より小さいユニットに再分割され得る。ＲＱＴのリーフノードは変換ユニット（ＴＵ）と呼ばれることがある。ＴＵに関連するピクセル差分値は、変換係数を生成するために変換され得、その変換係数は量子化され得る。
【００３２】
[0043] リーフＣＵは１つまたは複数の予測ユニット（ＰＵ）を含み得る。概して、ＰＵは、対応するＣＵの全部または一部分に対応する空間エリアを表し、そのＰＵの参照サンプルを取り出しおよび／または生成するためのデータを含み得る。その上、ＰＵは、予測に関係するデータを含む。たとえば、ＰＵがイントラモード符号化されるとき、ＰＵのためのデータは、ＰＵに対応するＴＵのためのイントラ予測モードを記述するデータを含み得る残差４分木（ＲＱＴ）中に含まれ得る。ＲＱＴは変換ツリーと呼ばれることもある。いくつかの例では、イントラ予測モードは、ＲＱＴの代わりに、リーフＣＵシンタックス中でシグナリングされ得る。別の例として、ＰＵがインターモード符号化されるとき、ＰＵは、ＰＵのための、１つまたは複数の動きベクトルなど、動き情報を定義するデータを含み得る。ＰＵのための動きベクトルを定義するデータは、たとえば、動きベクトルの水平成分、動きベクトルの垂直成分、動きベクトルについての解像度（たとえば、１／４ピクセル精度または１／８ピクセル精度）、動きベクトルが指す参照ピクチャ、および／または動きベクトルのための参照ピクチャリスト（たとえば、リスト０、リスト１、またはリストＣ）を記述し得る。
【００３３】
[0044] １つまたは複数のＰＵを有するリーフＣＵはまた、１つまたは複数の変換ユニット（ＴＵ）を含み得る。変換ユニットは、上記で説明されたように、（ＴＵ４分木構造とも呼ばれる）ＲＱＴを使用して指定され得る。たとえば、スプリットフラグは、リーフＣＵが４つの変換ユニットにスプリットされるかどうかを示し得る。次いで、各変換ユニットは、さらなるサブＴＵにさらにスプリットされ得る。ＴＵがさらにスプリットされないとき、そのＴＵはリーフＴＵと呼ばれることがある。概して、イントラコーディングの場合、リーフＣＵに属するすべてのリーフＴＵは同じイントラ予測モードを共有する。すなわち、概して、リーフＣＵのすべてのＴＵの予測値を計算するために同じイントラ予測モードが適用される。イントラコーディングでは、ビデオエンコーダは、イントラ予測モードを使用して各リーフＴＵの残差値を、ＴＵに対応するＣＵの一部と元のブロックとの間の差分として計算し得る。ＴＵは、必ずしもＰＵのサイズに制限されるとは限らない。したがって、ＴＵは、ＰＵよりも大きいことも小さいこともある。イントラコーディングでは、ＰＵは、同じＣＵのための対応するリーフＴＵとコロケート（collocate）され得る。いくつかの例では、リーフＴＵの最大サイズは、対応するリーフＣＵのサイズに対応し得る。
【００３４】
[0045] その上、リーフＣＵのＴＵはまた、残差４分木（ＲＱＴ）と呼ばれる、それぞれの４分木データ構造に関連し得る。すなわち、リーフＣＵは、リーフＣＵがどのようにＴＵに区分されるかを示す４分木を含み得る。ＴＵ４分木のルートノードは概してリーフＣＵに対応し、ＣＵ４分木のルートノードは概してツリーブロック（またはＬＣＵ）に対応する。スプリットされないＲＱＴのＴＵはリーフＴＵと呼ばれる。概して、本開示では、別段に記載されていない限り、リーフＣＵおよびリーフＴＵに言及するためにそれぞれＣＵおよびＴＵという用語を使用する。
【００３５】
[0046] ビデオシーケンスは、一般に、一連のビデオフレームまたはピクチャを含む。

ピクチャグループ（ＧＯＰ：group of pictures）は、概して、ビデオピクチャのうちの一連の１つまたは複数を備える。ＧＯＰは、ＧＯＰ中に含まれるいくつかのピクチャを記述するシンタックスデータを、ＧＯＰのヘッダ中、ピクチャのうちの１つまたは複数のヘッダ中、または他の場所に含み得る。ピクチャの各スライスは、それぞれのスライスのための符号化モードを記述するスライスシンタックスデータを含み得る。ビデオエンコーダ２０は、一般に、ビデオデータを符号化するために個々のビデオスライス内のビデオブロックに対して動作する。ビデオブロックはＣＵ内のコーディングノードに対応し得る。ビデオブロックは、固定サイズまたは可変サイズを有し得、指定されたコーディング規格に応じてサイズが異なり得る。

【００３６】

[0047] 一例として、予測は様々なサイズのＰＵについて実行され得る。特定のＣＵのサイズが２Ｎ×２Ｎであると仮定すると、イントラ予測が、２Ｎ×２ＮまたはＮ×ＮのＰＵサイズに対して実行され得、インター予測が、２Ｎ×２Ｎ、２Ｎ×Ｎ、Ｎ×２Ｎ、またはＮ×Ｎの対称的なＰＵサイズに対して実行され得る。インター予測のための非対称区分は、２Ｎ×ｎＵ、２Ｎ×ｎＤ、ｎＬ×２Ｎ、およびｎＲ×２ＮのＰＵサイズについても実行され得る。非対称区分では、ＣＵの一方向は区分されないが、他の方向は２５％と７５％とに区分される。２５％の区分に対応するＣＵの部分は、「ｎ」とその後ろに付く「Ｕｐ」、「Ｄｏｗｎ」、「Ｌｅｆｔ」、または「Ｒｉｇｈｔ」という表示によって示される。したがって、たとえば、「２Ｎ×ｎＵ」は、上部の２Ｎ×０．５Ｎ　ＰＵと下部の２Ｎ×１．５Ｎ　ＰＵとで水平方向に区分された２Ｎ×２Ｎ　ＣＵを指す。

【００３７】

[0048] 本開示では、「Ｎ×Ｎ（NxN）」および「Ｎ×Ｎ（N by N）」は、垂直寸法および水平寸法に関するビデオブロックのピクセル寸法、たとえば、１６×１６（16x16）ピクセルまたは１６×１６（16 by 16）ピクセルを指すために互換的に使用され得る。概して、１６×１６ブロックは、垂直方向に１６ピクセルを有し（ｙ＝１６）、水平方向に１６ピクセルを有する（ｘ＝１６）。同様に、Ｎ×Ｎブロックは、概して、垂直方向にＮピクセルを有し、水平方向にＮピクセルを有し、ここで、Ｎは非負整数値を表す。ブロック中のピクセルは行および列に配列され得る。その上、ブロックは、必ずしも、水平方向において垂直方向と同じ数のピクセルを有する必要があるとは限らない。たとえば、ブロックはＮ×Ｍピクセルを備え得、ここで、Ｍは必ずしもＮに等しいとは限らない。

【００３８】

[0049] ＣＵのＰＵを使用したイントラ予測コーディングまたはインター予測コーディングの後に、ビデオエンコーダ２０は、ＣＵのＴＵのための残差データを計算し得る。ＰＵは、（ピクセル領域とも呼ばれる）空間領域において予測ピクセルデータを生成する方法またはモードを記述するシンタックスデータを備え得、ＴＵは、変換、たとえば、残差ビデオデータへの離散コサイン変換（ＤＣＴ）、整数変換、ウェーブレット変換、または概念的に同様の変換の適用後に、変換領域において係数を備え得る。残差データは、符号化されていないピクチャのピクセルと、ＰＵに対応する予測値との間のピクセル差分に対応し得る。ビデオエンコーダ２０は、ＣＵのための残差データを表す量子化された変換係数を含むようにＴＵを形成し得る。すなわち、ビデオエンコーダ２０は、（残差ブロックの形態の）残差データを計算し、変換係数のブロックを生成するために残差ブロックを変換し、次いで、量子化された変換係数を形成するために変換係数を量子化し得る。ビデオエンコーダ２０は、量子化された変換係数を含むＴＵ、ならびに他のシンタックス情報（たとえば、ＴＵのためのスプリッティング情報）を形成し得る。

【００３９】

[0050] 上述のように、変換係数を生成するための任意の変換の後に、ビデオエンコーダ２０は、変換係数の量子化を実行し得る。量子化は、概して、係数を表すために使用されるデータの量をできるだけ低減するために変換係数が量子化され、さらなる圧縮を行うプロセスを指す。量子化プロセスは、係数の一部または全部に関連するビット深度（bit depth）を低減し得る。たとえば、量子化中にｎビット値がｍビット値に切り捨てられ得

、ここで、ｎはｍよりも大きい。

【００４０】

[0051] 量子化の後に、ビデオエンコーダは、変換係数を走査し、量子化された変換係数を含む２次元行列から１次元ベクトルを生成し得る。走査は、アレイの前部により高いエネルギー（したがって、より低い周波数）係数を配置し、アレイの後部により低いエネルギー（したがって、より高い周波数）係数を配置するように設計され得る。いくつかの例では、ビデオエンコーダ２０は、エントロピー符号化（entropy encode）され得るシリアル化ベクトルを生成するために、量子化された変換係数を走査するためにあらかじめ定義された走査順序を利用し得る。他の例では、ビデオエンコーダ２０は適応型走査を実行し得る。１次元ベクトルを形成するために、量子化された変換係数を走査した後に、ビデオエンコーダ２０は、たとえば、本開示で説明されるコンテキスト適応型バイナリ算術コーディング（ＣＡＢＡＣ）設計に従って、１次元ベクトルをエントロピー符号化し得る。ビデオエンコーダ２０はまた、ビデオデータを復号する際にビデオデコーダ３０が使用するための符号化ビデオデータに関連するシンタックス要素をエントロピー符号化し得る。

【００４１】

[0052] 概して、ビデオデコーダ３０は、符号化データを復号するためにビデオエンコーダ２０によって実行されるものと、相反するが、実質的に同様のプロセスを実行する。たとえば、ビデオデコーダ３０は、残差ブロックを再生するために、受信されたＴＵの係数を逆量子化および逆変換する。ビデオデコーダ３０は、予測されたブロックを形成するために、シグナリングされた予測モード（イントラ予測またはインター予測）を使用する。次いで、ビデオデコーダ３０は、元のブロックを再生するために、（ピクセルごとに）予測されたブロックと残差ブロックとを組み合わせる。ブロック境界に沿って視覚的アーティファクト（visual artifact）を低減するためにデブロッキングプロセスを実行することなど、追加の処理が実行され得る。さらに、ビデオデコーダ３０は、ビデオエンコーダ２０のＣＡＢＡＣ符号化プロセスに相反するが、それと実質的に同様の様式でＣＡＢＡＣを使用してシンタックス要素を復号し得る。

【００４２】

[0053] 本開示は、概して、ビデオエンコーダ２０が、ある情報をビデオデコーダ３０などの別のデバイスに「シグナリング（signaling）」することに言及することがある。ただし、ビデオエンコーダ２０は、いくつかのシンタックス要素をビデオデータの様々な符号化部分に関連付けることによって情報をシグナリングし得ることを理解されたい。すなわち、ビデオエンコーダ２０は、いくつかのシンタックス要素をビデオデータの様々な符号化部分のヘッダに記憶することによってデータを「シグナリング」し得る。いくつかの場合には、そのようなシンタックス要素は、ビデオデコーダ３０によって受信され、復号されるより前に、符号化され、記憶され（たとえば、ストレージデバイス３２に記憶され）得る。したがって、「シグナリング」という用語は、通信がリアルタイムまたはほぼリアルタイムで行われるか、あるいは、符号化時にシンタックス要素を媒体に記憶し、次いで、この媒体に記憶された後の任意の時間にそのシンタックス要素が復号デバイスによって取り出され得るときなどに行われ得る、ある時間期間にわたって行われるかどうかにかかわらず、概して、圧縮ビデオデータを復号するためのシンタックスまたは他のデータの通信を指し得る。

【００４３】

[0054] 以下のセクションは、ＢＡＣおよびＣＡＢＡＣ技法についてより詳細に説明する。ＢＡＣは、概して、再帰的間隔再分割プロシージャ（recursive interval-subdividing procedure）である。ＢＡＣは、Ｈ．２６４／ＡＶＣおよびＨ．２６５／ＨＥＶＣビデオコーディング規格におけるＣＡＢＡＣプロセスにおいてビンを符号化するために使用される。ＢＡＣコーダの出力は、最終コード化確率間隔（final coded probability interval）内の確率に対する値またはポインタを表すバイナリストリームである。確率間隔（probability interval）は、範囲（range）および下端値（lower end value）によって指定される。範囲は確率間隔の拡張である。低はコーディング間隔の下限である。

【００４４】
[0055] ビデオコーディングへの算術コーディングの適用は、Ｄ．Ｍａｒｐｅ、Ｈ．Ｓｃｈｗａｒｚ、およびＴ．Ｗｉｅｇａｎｄ「Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard」、ＩＥＥＥ Ｔｒａｎｓ．Ｃｉｒｃｕｉｔｓ ａｎｄ Ｓｙｓｔｅｍｓ ｆｏｒ Ｖｉｄｅｏ Ｔｅｃｈｎｏｌｏｇｙ、ｖｏｌ．１３、ｎｏ．７、２００３年７月に記載されている。ＣＡＢＡＣは、３つの主要な機能、すなわち、２値化（binarization）、コンテキストモデリング（context modeling）、および算術コーディング（arithmetic coding）を伴う。２値化は、シンタックス要素をバイナリシンボル（または「ビン」）にマッピングする機能を指す。バイナリシンボルは「ビンストリング（bin string）」と呼ばれることもある。コンテキストモデリングは、様々なビンの確率を推定する機能を指す。算術コーディングは、推定された確率に基づいて、ビンをビットに圧縮する後続の機能を指す。バイナリ算術コーダなど、様々なデバイスおよび／またはそれらのモジュールは算術コーディングの機能を実行し得る。
【００４５】
[0056] ＨＥＶＣでは、単項（Ｕ）、短縮単項（ＴＵ：truncated unary）、ｋ次指数ゴロム（ＥＧｋ：kth-order Exp-Golomb）、および固定長（ＦＬ：fixed length）を含む、いくつかの異なる２値化プロセスが使用される。様々な２値化プロセスの詳細は、Ｖ．ＳｚｅおよびＭ．Ｂｕｄａｇａｖｉ、「High throughput CABAC entropy coding in HEVC」、ＩＥＥＥ Ｔｒａｎｓａｃｔｉｏｎｓ ｏｎ Ｃｉｒｃｕｉｔｓ ａｎｄ Ｓｙｓｔｅｍｓ ｆｏｒ Ｖｉｄｅｏ Ｔｅｃｈｎｏｌｏｇｙ（ＴＣＳＶＴ）、ｖｏｌ．２２、ｎｏ．１２、１７７８～１７９１ページ、２０１２年１２月に記載されている。
【００４６】
[0057] ＣＡＢＡＣにおける各コンテキスト（すなわち、確率モデル）は状態によって表される。各状態（σ）は、特定のシンボル（たとえば、ビン）が劣勢シンボル（ＬＰＳ：Least Probable Symbol）である確率（p_σ）を暗黙的に表す。シンボルはＬＰＳまたは優勢シンボル（ＭＰＳ：Most Probable Symbol）であり得る。シンボルはバイナリであり、したがって、ＭＰＳおよびＬＰＳは０または１であり得る。確率は、対応するコンテキストについて推定され、算術コーダ（arithmetic coder）を使用してシンボルをエントロピーコーディングするために（暗黙的に）使用される。
【００４７】
[0058] ＢＡＣのプロセスは、コーディングすべきコンテキストとコーディングされているビンの値とに応じて、それの内部値「範囲」および「低」を変更する状態機械によって扱われる。コンテキストの状態（すなわち、それの確率）に応じて、範囲は、範囲MPS_σ（状態$_\sigma$における優勢シンボルの範囲）と範囲LPS_σ（状態$_\sigma$における劣勢シンボルの範囲）とに分割される。理論上、確率状態$_\sigma$の範囲LPS_σ値は以下の乗算によって導出される。
【００４８】
【数１】

$$\text{範囲LPS}_\sigma = \text{範囲} \times p_\sigma$$

【００４９】
上式で、p_σは、ＬＰＳを選択する確率である。もちろん、ＭＰＳの確率は$1-p_\sigma$である。等価的に、範囲MPS_σは、範囲－範囲LPS_σに等しい。ＢＡＣは、コーディングすべきコンテキストビンの状態と、現在の範囲と、コーディングされているビンの値（すなわち、ビンがＬＰＳに等しいのかＭＰＳに等しいのか）とに応じて、範囲を反復的に更新する。
【００５０】
[0059] 図２Ａおよび図２Ｂは、ビンｎにおけるこのプロセスの例を示す。図２Ａの例１００では、ビンｎにおいて、ビン２における範囲（range）は、あるコンテキスト状態（$_\sigma$）を仮定すると、ＬＰＳ（p_σ）の確率によって与えられる範囲ＭＰＳと範囲ＬＰＳとを含む。例１００は、ビンｎの値がＭＰＳに等しいときのビンｎ＋１における範囲の更

新を示す。この例では、低（low）は同じままであるが、ビンｎ＋１における範囲の値は、ビンｎにおける範囲ＭＰＳの値に低減される。図２Ｂの例１０２は、ビンｎの値がＭＰＳに等しくない（すなわち、ＬＰＳに等しい）ときのビンｎ＋１における範囲の更新を示す。この例では、低は、ビンｎにおける範囲ＬＰＳの下側範囲値に移動される。さらに、ビンｎ＋１における範囲の値は、ビンｎにおける範囲ＬＰＳの値に低減される。

【００５１】

[0060] ＨＥＶＣでは、範囲は９ビットで表され、低は１０ビットで表される。範囲値および低値を十分な精度で維持するための再正規化プロセス（renormalization process）がある。範囲が２５６よりも小さいときはいつでも、再正規化が行われる。したがって、範囲は、再正規化の後、常に２５６に等しいかまたはそれよりも大きい。範囲の値と低の値とに応じて、ＢＡＣは、ビットストリームに「０」または「１」を出力するか、または将来の出力のために保持するために（ＢＯ：未解決ビット（bits-outstanding）と呼ばれる）内部変数を更新する。図３は、範囲に応じたＢＡＣ出力の例を示す。たとえば、範囲および低が、あるしきい値（たとえば、５１２）を上回るとき、ビットストリームに「１」が出力される。範囲および低が、あるしきい値（たとえば、５１２）を下回るとき、ビットストリームに「０」が出力される。範囲および下側が、あるしきい値間にあるとき、ビットストリームに何も出力されない。代わりに、ＢＯ値が増分され、次のビンが符号化される。

【００５２】

[0061] ＨＥＶＣのＣＡＢＡＣコンテキストモデルでは、１２８個の状態がある。０から６３までであり得る（状態σによって示される）６４個の可能なＬＰＳ確率がある。各ＭＰＳは０または１であり得る。したがって、１２８個の状態は、６４個の状態確率×ＭＰＳのための２個の可能な値（０または１）である。したがって、確率モデルは７ビットエントリとして記憶され得る。各７ビットエントリにおいて、確率状態を表すために６ビットが割り振られ得、適用可能なコンテキストメモリ中の優勢シンボル（ＭＰＳ）のために１ビットが割り振られ得る。

【００５３】

[0062] ＬＰＳ範囲（範囲LPS_σ）を導出する計算を低減するために、ＨＥＶＣでは、すべての場合についての結果が事前計算され、近似値としてルックアップテーブルに記憶される。したがって、ＬＰＳ範囲は、単純なテーブルルックアップを使用することによって、乗算なしに取得され得る。乗算は、多くのハードウェアアーキテクチャにおいて有意なレイテンシ（latency）を生じ得るので、この動作を回避することは、いくつかのデバイスまたはアプリケーションにとって重要であり得る。

【００５４】

[0063] ４列事前計算ＬＰＳ範囲テーブルが乗算の代わりに使用され得る。範囲は４つのセグメントに分割される。セグメントインデックスは、質問（範囲＞＞６）＆３によって導出され得る。事実上、セグメントインデックスは、実際の範囲からビットをシフトし、ドロップすることによって導出される。以下の表１は、可能な範囲およびそれらの対応するインデックスを示す。

【００５５】

【表１】

表１－範囲インデックス

範囲	256-319	320-383	384-447	448-511
(範囲>>6) & 3	0	1	2	3

【００５６】

[0064] ＬＰＳ範囲テーブルは、次いで、６４個のエントリ（確率状態ごとに１つ）×４（範囲インデックスごとに１つ）を有する。各エントリは、範囲ＬＰＳ、すなわち、範囲にＬＰＳ確率を乗算した値である。このテーブルの部分の一例が以下の表２に示される

。表２は、確率状態９～１２を示す。ＨＥＶＣのための１つの提案では、確率状態は０～６３にわたり得る。

【００５７】

【表２】

表2－範囲LPS

確率状態（σ）	範囲LPS インデックス0	インデックス	インデックス2	インデックス3
…	…	…	…	…
9	90	110	130	150
10	85	104	123	142
11	81	99	117	135
12	77	94	111	128
…	…	…	…	…

【００５８】

[0065] 各セグメント（すなわち、範囲値）において、各確率状態σのＬＰＳ範囲があらかじめ定義される。言い換えれば、確率状態σのＬＰＳ範囲が４つの値（すなわち、範囲インデックスごとに１つの値）に量子化される。所与のポイントにおいて使用される特定のＬＰＳ範囲は、範囲がどのセグメントに属するかに依存する。テーブル中で使用される可能なＬＰＳ範囲の数は、テーブル列の数（すなわち、可能なＬＰＳ範囲値の数）とＬＰＳ範囲精度との間のトレードオフである。概して、より多数の列は、ＬＰＳ範囲値のより小さい量子化誤差を生じるが、また、テーブルを記憶するためにより多くのメモリの必要を増加させる。より少数の列は、量子化誤差を増加させるが、また、テーブルを記憶するために必要とされるメモリを低減する。

【００５９】

[0066] 上記で説明されたように、各ＬＰＳ確率状態は、対応する確率を有する。ＨＥＶＣでは、６４個の代表的確率値$p_\sigma \in [0.01875, 0.5]$が、再帰的式である以下の式（１）に従ってＬＰＳ（劣勢シンボル）について導出される。

【００６０】

【数２】

$$p_\sigma = \alpha * p_{\sigma-1} \text{ すべての } \sigma = 1,\cdots,63 \text{ である場合}$$

$$\text{ここで } \alpha = \left(\frac{0.01875}{0.5}\right)^{1/63}$$

【００６１】

[0067] 上記の例では、選定されたスケーリングファクタ（scaling factor）α ０．９４９２と確率のセットのカーディナリティ（cardinality）Ｎ＝６４の両方が、確率表現の精度と高速適応のための要望との間の良好な妥協を表す。いくつかの例では、１により近いαの値は、より高い精度をもつ遅い適応（「定常状態挙動（steady-state behavior）」）を生じ得、より速い適応は、精度の低減という犠牲を払ってαの値を減少させる、非定常の場合に達成され得る。スケーリングファクタαは、現在の更新に有意な影響を有する、前に符号化されたビンの数を示すウィンドウサイズに対応し得る。ＭＰＳ（優勢シンボル）の確率は、１－ＬＰＳ（劣勢シンボル）の確率に等しい。言い換えれば、ＭＰＳの確率は、式（１－ＬＰＳ）によって表され得、ここで、「ＬＰＳ」はＬＰＳの確率を表す。したがって、ＨＥＶＣにおけるＣＡＢＡＣによって表され得る確率範囲は、［０．０１８７５，０．９８１２５（＝１－０．０１８７５）］である。

【００６２】

[0068] シンタックス要素のための値のビット（または「ビン（bin）」）をコーディ

ングするために使用されるコンテキストの確率状態が、信号統計値（すなわち、たとえばシンタックス要素のために、前にコーディングされたビンの値）に従うために更新されるので、ＣＡＢＡＣは適応型である。更新プロセスは以下の通りである。所与の確率状態について、更新は、状態インデックスと、ＬＰＳまたはＭＰＳのいずれかとして識別される符号化シンボルの値とに依存する。更新プロセスの結果として、潜在的に修正されたＬＰＳ確率推定値と、必要な場合、修正されたＭＰＳ値とを含む、新しい確率状態が導出される。

【００６３】

[0069] 各ビンのコーディングの後にコンテキスト切替えが行われ得る。ＭＰＳに等しいビン値の場合、所与の状態インデックスは単に１だけ増分される。これは、ＬＰＳ確率がすでにそれの最小値にある（または、等価的に、最大ＭＰＳ確率が達せられる）、状態インデックス６２においてＭＰＳが発生したときを除く、すべての状態について。この場合、ＬＰＳが参照されるか、または最後のビン値が符号化されるまで、状態インデックスは固定のままである（最後のビン値の特殊な場合には特殊終了状態が使用される）。ＬＰＳが発生したとき、状態インデックスは、以下の式に示すように、状態インデックスをある量だけ減分することによって変更される。このルールは、概して、以下の例外とともにＬＰＳの各発生に適用される。ＬＰＳが、同程度の確率がある場合に対応する、インデックスσ＝０をもつ状態において符号化されたと仮定すると、状態インデックスは固定のままであるが、ＭＰＳ値は、ＬＰＳの値とＭＰＳの値とが交換されるようにトグルされることになる。すべての他の場合では、たとえどのシンボルが符号化されたとしても、ＭＰＳ値は改変されない。概して、ビデオコーダは、所与のＬＰＳ確率p_{old}と、それの更新されたカウンターパート（counterpart）p_{new}との間の関係を示す、以下の式（２）に従って新しい確率状態を導出し得る。

【００６４】

【数３】

$$p_{new} = \begin{cases} \max(\alpha * p_{old}, p_{62}), & \text{MPSが発生した場合} \\ \alpha * p_{old} + (1-\alpha), & \text{LPSが発生した場合} \end{cases} \qquad (2)$$

【００６５】

[0070] 複雑さを低減するために、ビデオコーダは、すべての遷移ルールが、いくつかのエントリをそれぞれ有する高々２つのテーブルによって実現され得るように、ＣＡＢＡＣを実装し得る。一例として、すべての遷移ルールは、７ビット符号なし整数値の１２８個のエントリをそれぞれ有する高々２つのテーブルによって実現され得る（たとえば、以下の表３および表４）。別の例として、すべての遷移ルールは、６ビット符号なし整数値の６３個のエントリをそれぞれ有する高々２つのテーブルによって実現され得る（たとえば、ＨＥＶＣの表９－４１）。状態インデックスｉが与えられれば、更新した後に、ビデオコーダは、新しい状態インデックスとして、ＭＰＳ値がコーディングされるときにＴｒａｎｓＩｄｘＭＰＳ［ｉ]を定義し、またはＬＰＳ値がコーディングされるときにＴｒａｎｓＩｄｘＬＰＳ［ｉ]を定義し得る。

【００６６】

【表３】

表3

```
TransIdxMPS[ 128 ] =
{
 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49,
 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65,
 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81,
 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97,
 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113,
 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 124, 125, 126, 127
};
```

【００６７】

【表４】

表4

```
TransIdxLPS[ 128 ] =
{
 1, 0, 0, 1, 2, 3, 4, 5, 4, 5, 8, 9, 8, 9, 10, 11,
 12, 13, 14, 15, 16, 17, 18, 19, 18, 19, 22, 23, 22, 23, 24, 25,
 26, 27, 26, 27, 30, 31, 30, 31, 32, 33, 32, 33, 36, 37, 36, 37,
 38, 39, 38, 39, 42, 43, 42, 43, 44, 45, 44, 45, 46, 47, 48, 49,
 48, 49, 50, 51, 52, 53, 52, 53, 54, 55, 54, 55, 56, 57, 58, 59,
 58, 59, 60, 61, 60, 61, 60, 61, 62, 63, 64, 65, 64, 65, 66, 67,
 66, 67, 66, 67, 68, 69, 68, 69, 70, 71, 70, 71, 70, 71, 72, 73,
 72, 73, 72, 73, 74, 75, 74, 75, 74, 75, 76, 77, 76, 77, 126, 127
};
```

【００６８】

[0071] いくつかの例では、ビデオコーダは、所与の状態インデックスσについて、ＬＰＳが観測された場合に新しい更新された状態インデックスＴｒａｎｓＩｄｘＬＰＳ［σ］を決定する、単一のテーブルＴｒａｎｓＩｄｘＬＰＳを用いて状態遷移を決定し得る。ＭＰＳ駆動型遷移は、１の固定値による状態インデックスの単純な（飽和）増分によって取得され、更新された状態インデックスｍｉｎ（σ＋１，６２）を生じることができる。

【００６９】

[0072] 上記で説明されたように、コンテキストモデリング（context modeling）は、より高いコーディング効率を達成するための寄与ファクタである正確な確率推定（probability estimation）を与える。したがって、コンテキストモデリングは適応型プロセスである。異なるコンテキストが異なるビンについて使用され得、コンテキストの確率は、前にコーディングされたビンの値に基づいて更新され得る。同様の分布をもつビンは、しばしば同じコンテキストを共有する。各ビンのためのコンテキストは、シンタックス要素のタイプ、シンタックス要素中のビン位置（ｂｉｎＩｄｘ）、ルーマ／クロマ情報、隣接情報などに基づいて選択され得る。

【００７０】

[0073] 所与のスライスをコーディングする前に、１つまたは複数のあらかじめ定義された値に基づいて確率モデルが初期化される。たとえば、ｑｐによって示される入力量子化パラメータと、ｉｎｉｔＶａｌによって示されるあらかじめ定義された値とが与えられれば、（状態およびＭＰＳによって示される）確率モデルの７ビットエントリが以下の式（３）に従って導出され得る。

【００７１】

【数４】

$$
\begin{aligned}
qp &= \mathrm{Clip3}(0, 51, qp);\\
\text{傾斜} &= (\mathrm{initVal} >> 4) * 5 - 45;\\
\text{オフセット} &= ((\mathrm{initVal} \,\&\, 15) << 3) - 16;\\
\mathrm{initState} &= \min(\max(1, (((\text{傾斜} * qp) >> 4) + \text{オフセット})), 126);\\
\mathrm{MPS} &= (\mathrm{initState} >= 64);\\
\text{状態インデックス} &= ((\mathrm{mpState}\ ?\ (\mathrm{initState} - 64) : (63 - \mathrm{initState})) << 1) + \mathrm{MPS};
\end{aligned}
\qquad (3)
$$

【００７２】

[0074] 導出された状態インデックスはＭＰＳ情報を暗黙的に含む。より詳細には、状態インデックスが偶数値であるとき、ＭＰＳ値は０に等しい。逆に、状態インデックスが奇数値であるとき、ＭＰＳ値は１に等しい。「ｉｎｉｔＶａｌ」の値は、８ビット精度での［０，２５５]の範囲内にある。あらかじめ定義された値「ｉｎｉｔＶａｌ」はスライス依存である。言い換えれば、確率モデルのためのコンテキスト初期化パラメータの３つのセットは、それぞれＩ、Ｐ、およびＢスライス中で１つずつ使用される。このようにして、ＣＡＢＡＣを実行するように構成されたビデオ符号化デバイスは、３つの初期化テーブル間でこれらのスライスタイプ（slice type）のために選定することを可能にされ、したがって、異なるコーディングシナリオおよび／または異なるタイプのビデオコンテンツへのより良い適合が達成され得る。

【００７３】

[0075] ＨＥＶＣによれば、１つのＰ（またはＢ）スライスがＢ（またはＰ）スライスを用いて初期化されることを可能にするために、別のツールが適用され得る。逆に、そのツールは、１つのＢスライスがＰスライスを用いて初期化されることを可能にするために適用され得る。関係するシンタックス要素が、（ＨＥＶＣのセクション７．３．６．１に対応する）以下の表５で説明され、表５の後に、関係するセマンティクス（semantics）および復号プロセス（decoding process）が以下で説明される。

【００７４】

【表５】

表5

slice_segment_header( ) {	記述子
first_slice_segment_in_pic_flag	u(1)
if(nal_unit_type >= BLA_W_LP && nal_unit_type <= RSV_IRAP_VCL23)	
no_output_of_prior_pics_flag	u(1)
slice_pic_parameter_set_id	ue(v)
if(!first_slice_segment_in_pic_flag) {	
if(dependent_slice_segments_enabled_flag)	
dependent_slice_segment_flag	u(1)
slice_segment_address	u(v)

}	
if(!dependent_slice_segment_flag) {	
for(i = 0; i < num_extra_slice_header_bits; i++)	
slice_reserved_flag[i]	u(1)
slice_type	ue(v)
if(output_flag_present_flag)	
pic_output_flag	u(1)
if(separate_colour_plane_flag == 1)	
colour_plane_id	u(2)
if(nal_unit_type != IDR_W_RADL && nal_unit_type != IDR_N_LP) {	
slice_pic_order_cnt_lsb	u(v)
short_term_ref_pic_set_sps_flag	u(1)
if(!short_term_ref_pic_set_sps_flag)	
short_term_ref_pic_set(num_short_term_ref_pic_sets)	
else if(num_short_term_ref_pic_sets > 1)	
short_term_ref_pic_set_idx	u(v)
if(long_term_ref_pics_present_flag) {	
...	
}	
}	
if(sps_temporal_mvp_enabled_flag)	
slice_temporal_mvp_enabled_flag	u(1)
}	
if(sample_adaptive_offset_enabled_flag) {	
slice_sao_luma_flag	u(1)
slice_sao_chroma_flag	u(1)
}	
if(slice_type == P || slice_type == B) {	
num_ref_idx_active_override_flag	u(1)
if(num_ref_idx_active_override_flag) {	
num_ref_idx_l0_active_minus1	ue(v)
if(slice_type == B)	
num_ref_idx_l1_active_minus1	ue(v)
}	
if(lists_modification_present_flag && NumPocTotalCurr > 1)	
ref_pic_lists_modification( )	
if(slice_type == B)	
mvd_l1_zero_flag	u(1)
if(cabac_init_present_flag)	

cabac_init_flag	u(1)
if(slice_temporal_mvp_enabled_flag) {	
if(slice_type == B)	
collocated_from_l0_flag	u(1)
if((collocated_from_l0_flag && num_ref_idx_l0_active_minus1 > 0) || (!collocated_from_l0_flag && num_ref_idx_l1_active_minus1 > 0))	
collocated_ref_idx	ue(v)
}	
if((weighted_pred_flag && slice_type == P) || (weighted_bipred_flag && slice_type == B))	
pred_weight_table( )	
five_minus_max_num_merge_cand	ue(v)
}	
...	
byte_alignment( )	
}	

【００７５】

[0076] 表５のシンタックス要素のためのセマンティクスは以下のように定義され得る。

【００７６】

[0077] １に等しいｃａｂａｃ＿ｉｎｉｔ＿ｐｒｅｓｅｎｔ＿ｆｌａｇは、ＰＰＳを参照するスライスヘッダ（slice header）中にｃａｂａｃ＿ｉｎｉｔ＿ｆｌａｇが存在することを指定する。０に等しいｃａｂａｃ＿ｉｎｉｔ＿ｐｒｅｓｅｎｔ＿ｆｌａｇは、ＰＰＳを参照するスライスヘッダ中にｃａｂａｃ＿ｉｎｉｔ＿ｆｌａｇが存在しないことを指定する。

【００７７】

[0078] ｃａｂａｃ＿ｉｎｉｔ＿ｆｌａｇは、以下で説明される復号プロセスにおいて定義されている、コンテキスト変数のための初期化プロセスにおいて使用される初期化テーブルを決定するための方法を指定する。ｃａｂａｃ＿ｉｎｉｔ＿ｆｌａｇが存在しないとき、それは０に等しいと推論される。

【００７８】

[0079] 記述子：

[0080] ａｅ（ｖ）：コンテキスト適応型算術エントロピーコード化シンタックス要素（context-adaptive arithmetic entropy-coded syntax element）。

【００７９】

[0081] ｂ（８）：任意のパターンのビットストリング（bit string）（８ビット）を有するバイト。

【００８０】

[0082] ｆ（ｎ）：左ビットが先頭の、（左から右に）書き込まれたｎ個のビットを使用する固定パターンビットストリング。

【００８１】

[0083] ｓｅ（ｖ）：左ビットが先頭の、符号付き整数０次指数ゴロムコード化シンタックス要素（signed integer 0-th order Exp-Golomb-coded syntax element）。

【００８２】

[0084] ｕ（ｎ）：ｎ個のビットを使用する符号なし整数。シンタックステーブル中でｎが「ｖ」であるとき、ビット数は、他のシンタックス要素の値に依存する様式で変動する。

【００８３】
[0085] ｕｅ（ｖ）：左ビットが先頭の、符号なし整数０次指数ゴロムコード化シンタックス要素（unsigned integer 0-th order Exp-Golomb-coded syntax element）。
【００８４】
[0086] ＨＥＶＣの表９－４は、３つの初期化タイプの各々について、初期化が必要とされるコンテキストインデックス（ｃｔｘＩｄｘ）を与える。表９－４は、さらに、初期化のために必要とされるｉｎｉｔＶａｌｕｅの値を含むテーブル番号（ｃｔｘＴａｂｌｅ）を含む。ＰおよびＢスライスタイプの場合、ｉｎｉｔＴｙｐｅの導出は、ｃａｂａｃ＿ｉｎｉｔ＿ｆｌａｇシンタックス要素の値に依存する。ビデオコーダは、Ｔｔｈｅ可変ｉｎｉｔＴｙｐｅは、続く以下の擬似コードによって記述される動作を使用してとして導出されるを導出し得る。
【００８５】
【数５】

```
if( slice_type == I )
    initType = 0
else if( slice_type == P )
    initType = cabac_init_flag ? 2 : 1
else
    initType = cabac_init_flag ? 1 : 2
```

【００８６】
[0087] 新しい算術コーダが、Ａｌｓｈｉｎら、「Multi-parameter probability up-date for CABAC」、文書：ＪＣＴＶＣ－Ｆ２５４、ＪＣＴ－ＶＣ ｏｆ ＩＴＵ－Ｔ ＳＧ１６ ＷＰ３およびＩＳＯ／ＩＥＣ ＪＴＣ１／ＳＣ２９／ＷＧ１１、第６回会議：トリノ、イタリア、２０１１年７月１４～２２日（以下「ＪＣＴＶＣ－Ｆ２５４」）、およびＡｌｓｈｉｎら、「CE1 (subset B): Multi-parameter probability up-date for CABAC」、文書：ＪＣＴＶＣ－Ｇ７６４、ＪＣＴ－ＶＣ ｏｆ ＩＴＵ－Ｔ ＳＧ１６ ＷＰ３およびＩＳＯ／ＩＥＣ ＪＴＣ１／ＳＣ２９／ＷＧ１１、第７回会議：ジュネーブ、スイス、２０１１年１１月２１～３０日（以下「ＪＣＴＶＣ－Ｇ７６４」）に記載されている。ＪＣＴＶＣ－Ｆ２５４およびＪＣＴＶ－Ｇ７６４では、あらゆる確率が１から３２７６７までの整数として表した。したがって、すべての計算は１６ビット精度で行われる。ＡＶＣ ＣＡＢＡＣにおいて利用される、確率のためのルックアップテーブル（たとえば、上記で説明されたＴｒａｎｓＩｄｘＭＰＳおよびＴｒａｎｓＩｄｘＬＰＳ）および指数メッシュの代わりに、ＪＣＴＶＣ－Ｆ２５４およびＪＣＴＶ－Ｇ７６４において提案されたコーダは、確率更新のために、均一メッシュと、乗算のない式を用いた明示的計算とを利用する。
【００８７】
[0088] 確率p_iが、（たとえば、以下の式（４）によって示されるように）（たとえば、ｋが１５に等しい）０から2^kまでの整数P_iである確率インデックスによって表されると仮定する。
【００８８】
【数６】

$$p_i=P_i/2^k \qquad (4)$$

【００８９】
[0089] （たとえば、以下の式（５）によって示されるように）現代の算術コーデックにおける確率更新のための最も頻繁に使用される以下の式に従うこと。
【００９０】
【数７】

$$p_{new} = \alpha y+(1-\alpha)\, p_{old} \qquad (5)$$

【００９１】
[0090] 式（５）において、現在のシンボルが優勢シンボル（ＭＰＳ）と一致する場合、ｙは「０」に等しく、そうでない場合、ｙは「１」に等しい。この式（すなわち、式（５））は、劣勢シンボル（ＬＰＳ）の確率のための推定値を与える。上記の説明と同様に、パラメータαは、現在の更新に有意な影響を有する、前に符号化されたビンの数を示すウィンドウサイズに対応し得る。
【００９２】
[0091] ウィンドウサイズ（Ｗ）が２のべき乗（Ｗ＝１／2^{M}、Ｍは正の整数である）であると仮定した場合、式（４）におけるＰｉが入力p_{old}として与えられれば、更新された確率インデックスは、式（６）において以下で示される中で書き直され得る。
【００９３】
【数８】

$$P_i = ((2^k) >> M) + P_i - (P_i >> M) \quad (6)$$

【００９４】
[0092] ＪＣＴＶＣ－Ｆ２５４およびＪＣＴＶ－Ｇ７６４によって提案された１確率更新モデルでは、Ｍは、すべてのコンテキストについて固定であり、１つのレジスタのみが、更新された確率を記録するために使用される。一例では、Ｍは６に等しく設定される。すなわち、ウィンドウサイズは６４に等しい。確率更新プロセスは、以下の式（７）によって表され得る。
【００９５】
【数９】

$$P_{new} = ((2^k) >> 6) + P_i - (P_i >> 6) \quad (7)$$

【００９６】
[0093] ＪＣＴＶＣ－Ｆ２５４およびＪＣＴＶ－Ｇ７６４によって提案された技法の主要なアイデアは、異なるウィンドウサイズでの（ただ１つではなく）いくつかの確率推定を使用し、それらを次のビン確率予測のために加重平均として組み合わせることである。以下の式（８）および（９）は、ＪＣＴＶＣ－Ｆ２５４およびＪＣＴＶ－Ｇ７６４によって提案された技法の一例を示す。各確率p_iのための式（８）における計算は独立である。
【００９７】
【数１０】

$$p_{i\,new} = W_i\, y + (1 - W_i)\, p_{i\,old} \quad (8)$$

【００９８】
【数１１】

$$p_{new} = \Sigma \beta_i\, p_{i\,new} \quad (9)$$

【００９９】
[0094] 各確率p_iのための式（８）における計算は独立である。
【０１００】
[0095] ＪＣＴＶＣ－Ｆ２５４およびＪＣＴＶ－Ｇ７６４によって提案された方法では、確率推定のための線形結合は、式（１０）および（１１）に示されているように、W_0＝１６およびW_1＝２５６（W_i＝１／α_i）対応する２つの被加数からなる。式（１０）および（１１）において、最後のコーディングビンが「１」である場合、Ｙ＝2^{15}であり、最後のコーディングビンが「０」である場合、Ｙ＝０であり、「＞＞Ｍ」は、Ｍビットのための右算術シフトである。
【０１０１】
【数１２】

$$P_0 = (Y>>4) + P_0 - (P_0>>4) \quad (10)$$

【０１０２】

【数１３】

$$P_1 = (Y>>8) + P_1 - (P_0>>8) \quad (11)$$

【０１０３】

【数１４】

$$P = (P_0 + P_1+1)>>1 \quad (12)$$

【０１０４】

[0096] 短い遷移期間の場合、高速更新速度をもつ短距離予測（すなわち、より小さいウィンドウサイズ）のみが好ましい。しかし、最適値の近くでの安定化の後は、大部分のコンテキストについて２確率更新モデルがより正確である。ＪＣＴＶＣ－Ｆ２５４およびＪＣＴＶ－Ｇ７６４は、最後の初期化からの更新のカウンタを導入することを提案する。あらゆる更新の後に、カウンタは１だけ増加する。カウンタが何らかのしきい値を超えるまで、式（１０）によって定義される短い「ウィンドウサイズ」モデルのみが使用されることになる。カウンタがしきい値に達したとき、上記の式（１２）によって定義されるより正確な２確率更新モデルに切り替えるべきである。ＪＣＴＶＣ－Ｆ２５４およびＪＣＴＶ－Ｇ７６４によって提案された範囲計算プロセスは、５１２×６４ルックアップテーブルを用いて実行される。

【０１０５】

[0097] ＪＣＴＶＣ－Ｆ２５４およびＪＣＴＶ－Ｇ７６４によって提案された方法によれば、異なるコンテキスト初期化方法が適用される。詳細には、（それぞれ、ａｓＣｔｘＩｎｉｔ［０]およびａｓＣｔｘＩｎｉｔ［１]によって示される）２つのパラメータが、式（１３）に示されているように各コンテキストのためにあらかじめ定義される。

【０１０６】

【数１５】

```
Int iQPreper = 1 slice ? 37 : 40;
Int c=asCtxInit[ 0 ]+asCtxInit[ 1 ]*( iQp - iQPreper);
iP0= min( max( 1, c ), 32767 );
```

(13)

【０１０７】

[0098] １確率更新モデルでは、コンテキストは、１５ビット精度でｉＰ０によって表される。２確率更新モデルでは、別の変数ｉＰ１が最初にｉＰ０に等しく設定され、いくつのビンがコーディングされたかのカウンタがさらに必要とされる。ＪＣＴＶＣ－Ｆ２５４およびＪＣＴＶ－Ｇ７６４によって提案された方法では、ａｓＣｔｘＩｎｉｔ［０]とａｓＣｔｘＩｎｉｔ［１]の両方は１６ビットに記憶される。

【０１０８】

[0099] しかしながら、いくつかの例では、上記で説明された技法（すなわち、ＨＥＶＣのＣＡＢＡＣ技法、およびＪＣＴＶＣ－Ｆ２５４およびＪＣＴＶ－Ｇ７６４によって提案された修正）は、コーディング効率を低減し、および／またはコーダシステムリソースを準最適に利用し得る、１つまたは複数の問題を有し得る。

【０１０９】

[0100] 一例として、（たとえば、ＨＥＶＣまたはＨ．２６４／ＡＶＣにおいて使用される）上記で説明されたルックアップテーブルベースの算術コーダ技法では、確率更新は、固定ウィンドウサイズでの固定テーブル（すなわち、ＴｒａｎｓＩｄｘＬＰＳおよびＴｒａｎｓＩｄｘＭＰＳ）に基づく。固定ウィンドウサイズのこの使用により、更新速度が固定されることになる。しかしながら、シンタックス要素が発生し、コーディングされる必要がある頻度は、所与のＣＴＵまたはスライスについてまったく異なり得る。所与のＣＴＵまたはスライスについての異なる頻度で発生するシンタックス要素と組み合わせられた固定更新速度の制限により、より低い頻度で発生するシンタックス要素の推定された確率が準最適になり得る。たとえば、１つのＣＵについて、ｉｎｔｅｒ＿ｐｒｅｄ＿ｉｄｃの最高２つの値がシグナリングされ得、１つのＣＵ内の変換係数が数回コーディングされ得る。この場合、これらのシンタックス要素について同じ更新速度を使用するとき、１に等しいｉｎｔｅｒ＿ｐｒｅｄ＿ｉｄｃの推定された確率は、変換係数の確率が比較的最適

になっていることがあるにもかかわらず、１つのスライス全体をコーディングした後にまだ準最適であり得る。

【０１１０】

[0101] 別の例として、（たとえば、ＪＣＴＶＣ－Ｆ２５４およびＪＣＴＶ－Ｇ７６４によって提案された）カウンタ技法に基づく上記で説明された算術コーダでは、確率更新速度（probability update speed）は固定であり、高い精度（たとえば、可能な確率インデックスが［１，$2^{15}-1$］であり得るは、より低い頻度で選択されるシンタックス要素について低い効率を生じ、これは、望ましくないことがある。

【０１１１】

[0102] 別の例として、カウンタ技法に基づく算術コーダの２確率更新モデル構成要素では、２つのステータスパラメータ（確率インデックス）が記憶され、更新されなければならず、これは、ＣＡＢＡＣプロセスのスループットを望ましくなく制限し得る。

【０１１２】

[0103] また別の例として、画像／ビデオコーディングシステムでは、数百個のコンテキストが使用され得る。ＪＣＴＶＣ－Ｆ２５４およびＪＣＴＶ－Ｇ７６４によって提案された技法では、コンテキストごとに３２ビットが必要とされるが、ＨＥＶＣにおける算術コーダには８ビットのみで十分である。したがって、ＪＣＴＶＣ－Ｆ２５４およびＪＣＴＶ－Ｇ７６４によって提案された技法におけるコンテキスト初期化のためのあらかじめ定義された値のストレージは３００％だけ増加され、これは、ストレージに関するハードウェア実装形態について望ましくないことがある。

【０１１３】

[0104] 本開示の１つまたは複数の技法によれば、ビデオコーダ（たとえば、ビデオエンコーダ２０および／またはビデオデコーダ３０）は、異なるコンテキストについて異なるウィンドウサイズを使用し得る。たとえば、すべてのコンテキストについて固定ウィンドウサイズを使用するＨＥＶＣとは対照的に、ビデオコーダは、第１のコンテキストを更新するときに第１のウィンドウサイズを使用し、第２のコンテキストを更新するときに第２の異なるウィンドウサイズを使用し得る。いくつかの例では、ビデオコーダは、まれに使用されるコンテキストについて比較的より小さいウィンドウサイズを使用し得、頻繁に使用されるコンテキストについて比較的より大きいウィンドウサイズを使用し得る。コンテキストが使用される頻度によりぴったり適合されたウィンドウサイズを使用することによって、ビデオコーダは、すべてのコンテキストについて固定ウィンドウサイズを使用するそれ、精度と適応速度との間でより好都合に妥協しながらコンテキストを更新し得る。このようにして、本開示の技法はＣＡＢＡＣの効率を改善し得、これは、ビデオコーダが、ビデオデータを符号化するために必要とされるビット数を低減することを可能にし得る。

【０１１４】

[0105] 本開示で説明される技法は、たとえば、ビデオエンコーダ、ビデオデコーダ、または組み合わせられたビデオエンコーダデコーダ（ＣＯＤＥＣ）内で実行され得る。特に、そのような技法は、ビデオエンコーダのエントロピー符号化ユニット（entropy encoding unit）および／またはビデオデコーダのエントロピー復号ユニット（entropy decoding unit）において実行され得る。本技法は、たとえば、ＨＥＶＣ規格の態様によるビデオコーディングなど、ビデオコーディングをサポートするように構成され得るＣＡＢＡＣプロセス内で実行され得るエントロピー符号化および復号ユニットは、たとえば、残差ビデオデータに関連する量子化された変換係数、動きベクトル情報、シンタックス要素、ならびにビデオ符号化および／またはビデオ復号プロセスにおいて有用であり得る他のタイプの情報など、様々なビデオデータのうちのいずれかを符号化または復号するために、相反するまたは逆の様式でコーディングプロセスを適用するであり得る。

【０１１５】

[0106] 図４は、本開示で説明される、ＢＡＣコーディングのための技法を利用するように構成され得るビデオエンコーダ２０の一例を示すブロック図である。ビデオエンコー

ダ２０は、例示のためにＨＥＶＣコーディングのコンテキストにおいて説明されるが、他のコーディング規格または方法に関して本開示を限定するものではない。その上、ビデオエンコーダ２０は、ＨＥＶＣの範囲拡張（range extension）に従って技法を実装するように構成され得る。

【０１１６】

[0107] ビデオエンコーダ２０は、ビデオスライス内のビデオブロックのイントラコーディングおよびインターコーディングを実行し得る。イントラコーディングは、所与のビデオピクチャ内のビデオの空間冗長性を低減または除去するために空間予測に依拠する。インターコーディングは、ビデオシーケンスの隣接するピクチャ内のビデオの時間冗長性を低減または除去するか、あるいは他のビュー中のビデオに関する冗長性を低減または除去するために、時間予測またはビュー間予測に依拠する。

【０１１７】

[0108] 図４の例では、ビデオエンコーダ２０は、ビデオデータメモリ４０と、予測処理ユニット４２と、参照ピクチャメモリ６４と、加算器５０と、変換処理ユニット５２と、量子化処理ユニット５４と、エントロピー符号化ユニット（entropy encoding unit）５６とを含む。予測処理ユニット４２は、動き推定ユニット４４と、動き補償ユニット４６と、イントラ予測ユニット４８とを含む。ビデオブロック再構成のために、ビデオエンコーダ２０はまた、逆量子化処理ユニット５８と、逆変換処理ユニット６０と、加算器６２とを含む。再構成されたビデオからブロッキネスアーティファクト（blockiness artifact）を除去するためにブロック境界をフィルタ処理するための（図４に示されていない）デブロッキングフィルタも含まれ得る。所望される場合、デブロッキングフィルタは、一般に、加算器６２の出力をフィルタ処理することになる。（ループ中またはループ後の）追加のループフィルタもデブロッキングフィルタに加えて使用され得る。

【０１１８】

[0109] ビデオデータメモリ４０は、ビデオエンコーダ２０の構成要素によって符号化されるべきビデオデータを記憶し得る。ビデオデータメモリ４０に記憶されるビデオデータは、たとえば、ビデオソース１８から取得され得る。参照ピクチャメモリ６４は、（たとえば、イントラ予測コーディングモードまたはインター予測コーディングモードとも呼ばれる、イントラコーディングモードまたはインターコーディングモードで）ビデオエンコーダ２０によってビデオデータを符号化する際に使用するための参照ビデオデータを記憶する復号ピクチャバッファ（ＤＰＢ：decoded picture buffer）の一例である。ビデオデータメモリ４０および参照ピクチャメモリ６４は、同期ＤＲＡＭ（ＳＤＲＡＭ）を含むダイナミックランダムアクセスメモリ（ＤＲＡＭ）、磁気抵抗ＲＡＭ（ＭＲＡＭ）、抵抗性ＲＡＭ（ＲＲＡＭ（登録商標））、または他のタイプのメモリデバイスなど、様々なメモリデバイスのうちのいずれかによって形成され得る。ビデオデータメモリ４０および参照ピクチャメモリ６４は、同じメモリデバイスまたは別個のメモリデバイスによって与えられ得る。様々な例では、ビデオデータメモリ４０は、ビデオエンコーダ２０の他の構成要素とともにオンチップであるか、またはそれらの構成要素に対してオフチップであり得る。

【０１１９】

[0110] 符号化プロセス中に、ビデオエンコーダ２０は、コーディングされるべきビデオピクチャまたはスライスを受信する。ピクチャまたはスライスは複数のビデオブロックに分割され得る。動き推定ユニット４４および動き補償ユニット４６は、時間圧縮を行うかまたはビュー間圧縮を行うために、１つまたは複数の参照ピクチャ中の１つまたは複数のブロックに対する受信されたビデオブロックのインター予測コーディングを実行する。イントラ予測ユニット４８は、代替的に、空間圧縮を行うために、コーディングされるべきブロックと同じピクチャまたはスライス中の１つまたは複数の隣接ブロックに対する受信されたビデオブロックのイントラ予測コーディングを実行し得る。ビデオエンコーダ２０は、（たとえば、ビデオデータのブロックごとに適切なコーディングモードを選択するために）複数のコーディングパスを実行し得る。

【０１２０】
[0111] その上、パーティションユニット（図示せず）が、前のコーディングパスにおける前の区分方式の評価に基づいて、ビデオデータのブロックをサブブロックに区分し得る。たとえば、パーティションユニットは、初めにピクチャまたはスライスをＬＣＵに区分し、レートひずみ分析（たとえば、レートひずみ最適化）に基づいてＬＣＵの各々をサブＣＵに区分し得る。予測処理ユニット４２は、さらに、サブＣＵへのＬＣＵの区分を示す４分木データ構造を生成し得る。４分木のリーフノードＣＵは、１つまたは複数のＰＵと１つまたは複数のＴＵとを含み得る。
【０１２１】
[0112] 予測処理ユニット４２は、たとえば、誤差結果に基づいてコーディングモード、すなわち、イントラまたはインターのうちの１つを選択し得、残差ブロックデータを生成するために、得られたイントラコード化ブロックまたはインターコード化ブロックを加算器５０に与え、参照ピクチャとして使用するための符号化ブロックを再構成するために、得られたイントラコード化ブロックまたはインターコード化ブロックを加算器６２に与える。予測処理ユニット４２はまた、動きベクトル、イントラモードインジケータ、パーティション情報、および他のそのようなシンタックス情報など、シンタックス要素をエントロピー符号化ユニット５６に与える。
【０１２２】
[0113] 動き推定ユニット４４と動き補償ユニット４６とは、高度に統合され得るが、概念的な目的のために別々に示されている。動き推定ユニット４４によって実行される動き推定は、ビデオブロックの動きを推定する動きベクトルを生成するプロセスである。動きベクトルは、たとえば、現在ピクチャ（または他のコード化ユニット）内でコーディングされている現在ブロックに対する参照ピクチャ（または他のコード化ユニット）内の予測ブロックに対する現在ビデオピクチャ内のビデオブロックのＰＵの変位を示し得る。予測ブロックは、絶対差分和（ＳＡＤ：sum of absolute difference）、２乗差分和（ＳＳＤ：sum of square difference）、または他の差分メトリックによって決定され得るピクセル差分に関して、コーディングされるべきブロックにぴったり一致することがわかるブロックである。いくつかの例では、ビデオエンコーダ２０は、参照ピクチャメモリ６４に記憶された参照ピクチャのサブ整数ピクセル位置の値を計算し得る。たとえば、ビデオエンコーダ２０は、参照ピクチャの１／４ピクセル位置、１／８ピクセル位置、または他の分数ピクセル位置の値を補間し得る。したがって、動き推定ユニット４４は、フルピクセル位置と分数ピクセル位置とに対して動き探索を実行し、分数ピクセル精度で動きベクトルを出力し得る。
【０１２３】
[0114] 動き推定ユニット４４は、ＰＵの位置を参照ピクチャの予測ブロックの位置と比較することによって、インターコード化スライス中のビデオブロックのＰＵのための動きベクトルを計算する。参照ピクチャは、参照ピクチャメモリ６４に記憶された１つまたは複数の参照ピクチャを識別する１つまたは複数の参照ピクチャリスト（ＲＰＬ：reference picture list）から選択され得る。動き推定ユニット４４は、計算された動きベクトルをエントロピー符号化ユニット５６と動き補償ユニット４６とに送る。いくつかの例では、動き推定ユニット４４は、選択された参照ピクチャの指示をエントロピー符号化ユニット５６に送り得る。
【０１２４】
[0115] 動き補償ユニット４６によって実行される動き補償は、動き推定ユニット４４によって決定された動きベクトルに基づいて予測ブロックをフェッチまたは生成することを伴い得る。同じく、動き推定ユニット４４および動き補償ユニット４６は、いくつかの例では、機能的に統合され得る。現在ビデオブロックのＰＵのための動きベクトルを受信すると、動き補償ユニット４６は、動きベクトルが参照ピクチャリスト（ＲＰＬ）のうちの１つにおいて指す予測ブロックの位置を特定し得る。加算器５０は、以下で説明されるように、コーディングされている現在ブロックのピクセル値から予測ブロックのピクセル

値を減算し、ピクセル差分値を形成することによって、残差ビデオブロックを形成する。概して、動き推定ユニット４４はルーマ成分に対して動き推定を実行し、動き補償ユニット４６は、クロマ成分とルーマ成分の両方のためにルーマ成分に基づいて計算された動きベクトルを使用する。予測処理ユニット４２はまた、ビデオスライスのビデオブロックを復号する際にビデオデコーダ３０が使用するためのビデオブロックとビデオスライスとに関連するシンタックス要素を生成し得る。

【０１２５】

[0116] イントラ予測ユニット４８は、上記で説明されたように、動き推定ユニット４４と動き補償ユニット４６とによって実行されるインター予測の代替として、現在ブロックをイントラ予測し得る。特に、イントラ予測ユニット４８は、現在ブロックを符号化するために使用すべきイントラ予測モードを決定し得る。いくつかの例では、イントラ予測ユニット４８は、たとえば、別個の符号化パス中に、様々なイントラ予測モードを使用してブロックを符号化し得、イントラ予測ユニット４８は、複数のイントラ予測モードから使用するのに適切なイントラ予測モードを選択し得る。

【０１２６】

[0117] たとえば、イントラ予測ユニット４８は、様々なテストされたイントラ予測モードのためのレートひずみ分析（rate-distortion analysis）を使用してレートひずみ値を計算し、テストされたモードの中で最良のレートひずみ特性を有するイントラ予測モードを選択し得る。レートひずみ分析は、概して、符号化ブロックと、符号化ブロックを生成するために符号化された元の符号化されていないブロックとの間のひずみ（または誤差）の量、ならびに符号化ブロックを生成するために使用されるビットレート（すなわち、ビット数）を決定する。イントラ予測ユニット４８は、どのイントラ予測モードがブロックについて最良のレートひずみ値を呈するかを決定するために、様々な符号化ブロックのためのひずみおよびレートから比を計算し得る。いくつかの例では、複数のイントラ予測モードの各々は、イントラ予測ユニット４８によって（すなわち、ビデオデコーダに）シグナリングされ得る、対応するモードインデックスを有し得る。

【０１２７】

[0118] ビデオエンコーダ２０は、コーディングされている元のビデオブロックから、予測処理ユニット４２からの予測データを減算することによって残差ビデオブロックを形成する。加算器５０は、この減算演算を実行する１つまたは複数の構成要素を表す。

【０１２８】

[0119] 変換処理ユニット５２は、離散コサイン変換（ＤＣＴ）または概念的に同様の変換などの変換を残差ブロックに適用し、残差変換係数値を備えるビデオブロックを生成する。変換処理ユニット５２は、ＤＣＴと概念的に同様である他の変換を実行し得る。ウェーブレット変換、整数変換、サブバンド変換または他のタイプの変換も使用され得る。いずれの場合も、変換処理ユニット５２は、変換を残差ブロックに適用し、残差変換係数のブロックを生成する。変換は、残差情報をピクセル値領域から周波数領域などの変換領域に変換し得る。

【０１２９】

[0120] 変換処理ユニット５２は、得られた変換係数を量子化処理ユニット５４に送り得る。量子化処理ユニット５４は、ビットレートをさらに低減するために変換係数を量子化する。量子化プロセスは、係数の一部または全部に関連するビット深度を低減し得る。量子化の程度は、量子化パラメータ（quantization parameter）を調整することによって修正され得る。いくつかの例では、量子化処理ユニット５４は、次いで、量子化された変換係数を含む行列の走査を実行し得る。代替的に、エントロピー符号化ユニット５６が走査を実行し得る。

【０１３０】

[0121] 変換係数が１次元アレイに走査されると、エントロピー符号化ユニット５６は、コンテキスト適応型可変長コーディング（ＣＡＶＬＣ）、コンテキスト適応型バイナリ算術コーディング（ＣＡＢＡＣ）、確率間隔区分エントロピーコーディング（ＰＩＰＥ）

、ゴロムコーディング、ゴロム－ライスコーディング、指数ゴロムコーディング、シンタックスベース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）、または別のエントロピーコーディング方法などのエントロピーコーディングを係数に適用し得る。本開示の例による、様々な異なるエントロピーコーディングプロセスへの参照が行われるが、エントロピー符号化ユニット５６は、上記で説明されたようにＢＡＣコーディングを実行するように構成され得る。

【０１３１】

[0122] ＣＡＶＬＣを実行するために、エントロピー符号化ユニット５６は、送信されるべきシンボルのための可変長コードを選択し得る。ＶＬＣ中のコードワードは、比較的より短いコードが、可能性がより高いシンボルに対応し、より長いコードが、可能性がより低いシンボルに対応するように構成され得る。このようにして、ＶＬＣの使用は、たとえば、送信されるべき各シンボルのための等長コードワードを使用することに勝るビット節約を達成し得る。

【０１３２】

[0123] ＣＡＢＡＣを実行するために、エントロピー符号化ユニット５６は、送信されるべきシンボルを符号化するために、あるコンテキストに適用すべきコンテキストを選択し得る。コンテキストは、たとえば、隣接値が非０であるか否かに関係し得る。エントロピー符号化ユニット５６はまた、選択された変換を表す信号など、シンタックス要素をエントロピー符号化し得る。エントロピー符号化ユニット５６によるエントロピーコーディングの後に、得られた符号化ビデオは、ビデオデコーダ３０などの別のデバイスに送信されるか、あるいは後で送信するかまたは取り出すためにアーカイブされ得る。

【０１３３】

[0124] 本開示の１つまたは複数の技法によれば、エントロピー符号化ユニット５６は、ビデオデータを復号する際に、ビデオデコーダ３０など、ビデオデコーダが使用するためのデータ（たとえば、１次元バイナリベクトルとして表されるシンタックス要素値）をエントロピー符号化する（entropy code）とき、異なるウィンドウサイズを選択し得る。エントロピー符号化ユニット５６の一例のさらなる詳細は、図５を参照しながら以下で説明される。

【０１３４】

[0125] 逆量子化処理ユニット５８および逆変換処理ユニット６０は、たとえば、参照ブロックとして後で使用するために、ピクセル領域において残差ブロックを再構成するために、それぞれ逆量子化および逆変換を適用する。

【０１３５】

[0126] 動き補償ユニット４６はまた、動き推定において使用するためのサブ整数ピクセル値を計算するために、参照ブロックに１つまたは複数の補間フィルタを適用し得る。加算器６２は、参照ピクチャメモリ６４に記憶するための再構成されたビデオブロックを生成するために、動き補償ユニット４６によって生成された動き補償予測ブロックに再構成された残差ブロックを加算する。再構成されたビデオブロックは、後続のビデオピクチャ中のブロックをインターコーディングするために動き推定ユニット４４および動き補償ユニット４６によって参照ブロックとして使用され得る。現在ピクチャが現在ピクチャを予測するための参照ピクチャとして使用される場合など、いくつかの例では、動き補償ユニット４６および／または加算器６２は、現在ピクチャをコーディングしながら、一定の間隔で、参照ピクチャメモリ６４によって記憶された現在ピクチャのバージョンを更新し得る。一例として、動き補償ユニット４６および／または加算器６２は、現在ピクチャの各ブロックをコーディングした後に、参照ピクチャメモリ６４によって記憶された現在ピクチャのバージョンを更新し得る。たとえば、現在ブロックのサンプルが、初期化された値として参照ピクチャメモリ６４に記憶される場合、動き補償ユニット４６および／または加算器６２は、現在ブロックのための再構成されたサンプルで、参照ピクチャメモリ６４によって記憶された現在ピクチャの現在のサンプルを更新し得る。

【０１３６】

[0127] フィルタ処理ユニット（図示せず）は、様々なフィルタ処理プロセスを実行し得る。たとえば、フィルタ処理ユニットはデブロッキングを実行し得る。すなわち、フィルタ処理ユニットは、再構成されたビデオのスライスまたはフレームを形成する複数の再構成されたビデオブロックを受信し、スライスまたはフレームからブロッキネスアーティファクトを除去するために、ブロック境界をフィルタ処理し得る。一例では、フィルタ処理ユニットは、ビデオブロックのいわゆる「境界強度（boundary strength）」を評価する。ビデオブロックの境界強度に基づいて、ビデオブロックのエッジピクセルが、閲覧者にとって１つのビデオブロックからの遷移を知覚することがより困難になるように、隣接するビデオブロックのエッジピクセルに対してフィルタ処理され得る。

【０１３７】

[0128] いくつかの例では、動き補償ユニット４６および／または加算器６２は、フィルタ処理がサンプルに対するフィルタ処理（たとえば、デブロッキングおよび／またはＳＡＯ）を実行する前に、参照ピクチャメモリ６４によって記憶された現在ピクチャのバージョンを更新し得る。たとえば、フィルタ処理ユニットは、フィルタ処理を適用する前に、ピクチャ全体がコーディングされるまで待ち得る。このようにして、動き推定ユニット４４は、フィルタ処理を適用する前に、参照として現在ピクチャを使用し得る。いくつかの例では、フィルタ処理ユニットは、参照ピクチャメモリ６４によって記憶された現在ピクチャのバージョンが更新されたとき、フィルタ処理を実行し得る。たとえば、フィルタ処理ユニットは、各ブロックが更新されたとき、フィルタ処理を適用し得る。このようにして、動き推定ユニット４４は、フィルタ処理を適用した後、参照として現在ピクチャを使用し得る。

【０１３８】

[0129] 本技法のいくつかの異なる態様および例が本開示で説明されるが、本技法の様々な態様および例は、一緒にまたは互いに別々に実行され得る。言い換えれば、本技法は、上記で説明された様々な態様および例に厳密に限定されるべきではなく、組み合わせて使用されるか、あるいは一緒におよび／または別々に実行され得る。さらに、いくつかの技法は、（イントラ予測ユニット４８、動き補償ユニット４６、またはエントロピー符号化ユニット５６などの）ビデオエンコーダ２０のいくつかのユニットに起因され得るが、ビデオエンコーダ２０の１つまたは複数の他のユニットも、そのような技法を行うことを担当し得ることを理解されたい。

【０１３９】

[0130] 図５は、本開示の技法による、ＣＡＢＡＣを実行するように構成され得る例示的なエントロピー符号化ユニット（entropy encoding unit）５６のブロック図である。シンタックス要素１１８がエントロピー符号化ユニット５６に入力される。シンタックス要素がすでにバイナリ値シンタックス要素（たとえば、フラグ、または０および１の値のみを有する他のシンタックス要素）である場合、２値化のステップはスキップされ得る。シンタックス要素が非バイナリ値シンタックス要素（non-binary valued syntax element）（たとえば、１または０以外の値を有し得るシンタックス要素）である場合、非バイナリ値シンタックス要素はバイナライザ１２０によって２値化される。バイナライザ１２０は、バイナリ決定のシーケンスへの非バイナリ値シンタックス要素のマッピングを実行する。これらのバイナリ決定は、しばしば「ビン」と呼ばれる。たとえば、変換係数レベルでは、レベルの値は連続するビンに分けられ得、各ビンは、係数レベルの絶対値がある値よりも大きいか否かを示す。たとえば、（有意性フラグと呼ばれることがある）ビン０は、変換係数レベルの絶対値が０よりも大きいか否かを示す。ビン１は、変換係数レベルの絶対値が１よりも大きいか否かを示す、などである。各非バイナリ値シンタックス要素について、一意のマッピングが作成され得る。

【０１４０】

[0131] バイナライザ１２０によって生成された各ビンは、エントロピー符号化ユニット５６のバイナリ算術コーディング側に供給される。すなわち、非バイナリ値シンタックス要素の所定のセットについて、各ビンタイプ（たとえば、ビン０）が次のビンタイプ（

たとえば、ビン１）の前にコーディングされる。コーディングは、通常モードまたはバイパスモードのいずれかで実行され得る。バイパスモードでは、バイパスコーディングエンジン１２６が、固定確率モデルを使用して、たとえば、ゴロム－ライスまたは指数ゴロムコーディングを使用して、算術コーディングを実行する。バイパスモードは、概して、より予測可能なシンタックス要素のために使用される。

【０１４１】

[0132] 通常モードでのコーディングは、ＣＡＢＡＣを実行することを伴う。通常モードＣＡＢＡＣは、ビンの値の確率が、前にコーディングされたビンのそのときの値を与えられれば予測可能である場合に、ビン値をコーディングするためのものである。ビンがＬＰＳである確率がコンテキストモデラ（context modeler）１２２によって決定される。コンテキストモデラ１２２は、ビン値とコンテキストのための確率状態（たとえば、ＬＰＳの値と、ＬＰＳが発生する確率とを含む確率状態σ）とを出力する。コンテキストは、一連のビンのための初期コンテキストであり得るか、または前にコーディングされたビンのコード化値に基づいて決定され得る。上記で説明されたように、コンテキストモデラ１２２は、受信されたビンがＭＰＳであったのかＬＰＳであったのか否かに基づいて状態を更新し得る。コンテキストおよび確率状態 $_{\sigma}$ がコンテキストモデラ１２２によって決定された後、通常コーディングエンジン１２４がビン値に対してＢＡＣを実行する。

【０１４２】

[0133] 本開示の１つまたは複数の技法によれば、バイナリ算術コーディングプロセスにおいて確率状態を更新するために使用される変数の同じ値（たとえば、ウィンドウサイズ、またはスケーリングファクタ（α）、または固定確率更新速度のうちの１つまたは複数）を使用することとは対照的に、エントロピー符号化ユニット５６は、異なるコンテキストおよび／または異なるシンタックス要素について変数の異なる値を使用し得る。たとえば、コンテキストモデラ１２２は、複数のコンテキストのうちのコンテキストについて、バイナリ算術コーディングプロセスにおいて確率状態を更新するために使用される変数の値を決定し、決定された値に基づいて確率状態を更新し得る。

【０１４３】

[0134] いくつかの例では、次の確率状態を決定するためにコンテキストモデラ１２２によって使用されるウィンドウサイズはコンテキスト依存にされ得る。たとえば、コンテキストモデラ１２２は、異なるコンテキストについて異なるウィンドウサイズを使用し得る。一例として、コンテキストモデラ１２２は、複数のコンテキストのうちの第１のコンテキストのための第１のウィンドウサイズを決定し、第１のウィンドウサイズとは異なる、複数のコンテキストのうちの第２のコンテキストのための第２のウィンドウサイズを決定し得る。

【０１４４】

[0135] いくつかの例では、上記のコンテキスト依存更新方法を、ＪＣＴＶＣ－Ｆ２５４およびＪＣＴＶ－Ｇ７６４におけるものなど、カウンタベース算術コーダに組み込むとき、ウィンドウサイズの値はコンテキストに依存し得る。さらに、各コンテキストは、式（４）からの確率Ｐｉに加えて、ウィンドウサイズにさらに関連し得る。

【０１４５】

[0136] いくつかの例では、コンテキストモデラ１２２は、2^{M}に等しいことがあるウィンドウサイズWを使用し得、ここで、Mは正の整数であり得る。したがって、いくつかのコンテキストモデルは同じM値を有し得るが、各コンテキストは、他のコンテキストとは異なり得る、それ自体のM値を有し得る。

【０１４６】

[0137] いくつかの例では、コンテキストモデラ１２２は、ウィンドウサイズのあらかじめ定義されたセット（pre-defined set）からウィンドウサイズを決定し得る。いくつかの例示的なあらかじめ定義されたウィンドウサイズは、１６、３２、６４、および１２８であるが、他のウィンドウサイズが企図される。たとえば、可能なM値のセットがあらかじめ定義され得、たとえば、Mは、両端値を含む、４から７までにわたることがある。

いくつかの例では、コンテキストモデラ１２２は、可能なウィンドウサイズのセットの指示（たとえば、可能なM値のセットの指示）が、スライスヘッダ、あるいはピクチャパラメータセット、アクティブパラメータセット、シーケンスパラメータセット、またはビデオパラメータセットを含むパラメータセット中でシグナリングされることを引き起こし得る。

【０１４７】

[0138] いくつかの例では、各コンテキストに関連するウィンドウサイズ（たとえば、Mの値）はあらかじめ定義され得る。いくつかの例では、ウィンドウサイズは、さらに、スライスタイプおよび／または（たとえば、ＨＥＶＣではｔｅｍｐｏｒａｌＩｄと呼ばれる）時間識別子に依存し得る。いくつかの例では、ウィンドウサイズは、さらに、ピクチャタイプ（またはＮＡＬユニットタイプ）、たとえば、ピクチャがランダムアクセスピクチャであるか否かに依存し得る。

【０１４８】

[0139] いくつかの例では、コンテキストモデラ１２２は、各コンテキストに関連するウィンドウサイズ（たとえば、Mの値）が、スライスヘッダ／ピクチャパラメータセット／アクティブパラメータセット／シーケンスパラメータセット中でなど、ビットストリーム中でシグナリングされることを引き起こし得る。たとえば、各コンテキストのためのデフォルトウィンドウサイズ（default window size）が最初にあらかじめ定義され得る。各それぞれのコンテキストモデルについて、コンテキストモデラ１２２は、デフォルトウィンドウサイズがそれぞれのコンテキストのために使用されるかどうかを示す、それぞれのシンタックス要素（たとえば、フラグ）を符号化し得る。デフォルトウィンドウサイズがそれぞれのコンテキストのために使用されない場合、コンテキストモデラ１２２は、デフォルトウィンドウサイズに基づいて、実際の使用されるウィンドウサイズを差分符号化し得る。いくつかの例では、コンテキストモデラ１２２は、すべてのコンテキストの（すなわち、デフォルトウィンドウサイズが使用されるかどうかを示す）シンタックス要素を一緒に編成し、これらのシンタックス要素をコーディングするためにランレングスコーディング（run-length coding）を利用し得る。いくつかの例では、コンテキストモデラ１２２は、実際に使用されるウィンドウサイズとデフォルトウィンドウサイズとの間の差分をコーディングするとき、マッピングテーブル（mapping table）を利用し得る。たとえば、デフォルトのM値が６に等しい場合、可能なM値は、４、５、６、および７である。マッピングテーブルは次のように定義され得る。

【０１４９】

【表６】

実際のM値	4	5	6	7
コーディングされるべき値	0	1	-	2

【０１５０】

[0140] いくつかの例では、コンテキストモデラ１２２は、各コンテキストについて実際のウィンドウサイズとデフォルトウィンドウサイズとの間の差分を直接コーディングし得る。たとえば、デフォルトのM値が４である場合、コンテキストモデラ１２２は、各コンテキストについてM－４をコーディングし得る。

【０１５１】

[0141] いくつかの例では、コンテキストモデラ１２２は、現在スライス（current slice）中のコンテキストのためのすべてのウィンドウサイズが、前にコーディングされたスライス中の対応するコンテキストのための継承（inherit）されたウィンドウサイズである（すなわち、前にコーディングされたスライス中の対応するコンテキストのためのウィンドウサイズに等しく設定される）かどうかを示す、第１のシンタックス要素をコーディングし得る。一例では、「前に復号されたスライス（previously decoded slice）」は、現在スライスと同じスライスタイプ、または同じスライスタイプと量子化パラメータの両方、または同じスライスタイプと時間レイヤの両方、および／あるいは同じ初期化され

た量子化パラメータを有する、前にコーディングされたスライスとして定義され得る。いくつかの例では、前のスライスは、ＤＰＢ中に存在するピクチャに属することが必要であり得、参照ピクチャとして現在ピクチャのために使用され得、特に、ＨＥＶＣベースプラットフォームの場合のように、前のスライスは、参照ピクチャセット（ＲＰＳ）中のピクチャ、さらには、ＲＰＳの以下のサブセット、すなわち、ＲｅｆＰｉｃＳｅｔＳｔＣｕｒｒＢｅｆｏｒｅ、ＲｅｆＰｉｃＳｅｔＳｔＣｕｒｒＡｆｔｅｒ、およびＲｅｆＰｉｃＳｅｔＬｔＣｕｒｒのうちの１つ中のピクチャに属することが必要とされ得る。

【０１５２】

[0142] いくつかの例では、コンテキストモデラ１２２は、デフォルトウィンドウサイズが（たとえば、現在スライス中の）複数のコンテキストのために使用されるかどうかを示す第１のシンタックス要素をコーディングし得る。デフォルトウィンドウサイズが複数のコンテキストのために使用されない場合、コンテキストモデラ１２２は、コンテキストのためのウィンドウサイズを示す第２のシンタックス要素をコーディングし得る。たとえば、コンテキストモデラ１２２は、コンテキストのためのウィンドウサイズとデフォルトウィンドウサイズとの間の差分を示す第２のシンタックス要素をコーディングし得る。

【０１５３】

[0143] 別の例では、コンテキストモデラ１２２は、たとえば、前のスライスまたはピクチャからのコード化情報に基づいて、ウィンドウサイズを導出し得る。たとえば、コンテキストモデラ１２２は、１つのコンテキストに関連する前のスライス中のコーディングされたビンを追跡し得る。可能なウィンドウサイズの各候補について、コンテキストモデラ１２２は、これらのビンをコーディングするために消費されるビットを取得し、これらのビンをコーディングするための最小ビットを生じるウィンドウサイズを、このコンテキストのためのウィンドウサイズとして選択し得る。コンテキストモデラ１２２は、後続のスライス／ピクチャをコーディングするために、選択されたウィンドウサイズを使用し得る。

【０１５４】

[0144] いくつかの例では、算術コーダにおいて次の確率状態または確率更新速度（probability update speed）を決定するために使用される「ウィンドウサイズ（window size）」は、たとえば、コンテキストが、異なるシンタックス要素の間で共有される場合、シンタックス要素固有であり得る。たとえば、シンタックス要素のビンを符号化するためにコンテキストを使用するとき、コンテキストモデラ１２２は、シンタックス要素に基づいてコンテキストのためのウィンドウサイズを決定し得る。一例として、コーディングユニットスプリットシンタックス要素（coding unit split syntax element）のビンと、コーディングユニットスキップフラグシンタックス要素（coding unit skip flag syntax element）のビンとをコーディングするときの、コンテキストの状態を更新するために使用されるウィンドウサイズは同じ、たとえば、１６（すなわち、Ｍ＝４）であり得る。

【０１５５】

[0145] 本開示の１つまたは複数の技法によれば、コンテキストモデラ１２２は、ビデオデータを復号する際にビデオデコーダ３０が使用するためのデータ（たとえば、１次元ベクトルを表すシンタックス要素および／または他のシンタックス要素）をエントロピー符号化するとき、異なるウィンドウサイズを適応的に決定し得る。たとえば、各コンテキストについて、コンテキストモデラ１２２は、異なるウィンドウサイズで、記録されたビンストリングをコーディングするビットを計算し、最小ビットをもつ１つのウィンドウサイズを選択し得る。ウィンドウサイズがウィンドウサイズのあらかじめ定義されたセットから選択される場合、コンテキストモデラ１２２は、ウィンドウサイズのあらかじめ定義されたセットのうちのそれぞれのウィンドウサイズについて、コンテキストを用いてビンストリングを符号化するために使用されるビットのそれぞれの量を決定し、ビットの最も小さい量に対応する、ウィンドウサイズのあらかじめ定義されたセットのうちのウィンドウサイズを、そのコンテキストのためのウィンドウサイズとして選択し得る。

【０１５６】

[0146] いくつかの例では、上記の（１つまたは複数の）技法は特定のコンテキストに適用可能であり得る。すなわち、コンテキストのサブセットは、デフォルトウィンドウサイズではなく更新された「ウィンドウサイズ」を使用し得る。いくつかの例では、上記の（１つまたは複数の）技法は特定のスライスタイプに適用可能であり得る。

【０１５７】

[0147] 図４に戻ると、いくつかの場合には、エントロピー符号化ユニット５６またはビデオエンコーダ２０の別のユニットは、エントロピーコーディングに加えて、他のコーディング機能を実行するように構成され得る。たとえば、エントロピー符号化ユニット５６は、ＣＵとＰＵとのためのコード化ブロックパターン（ＣＢＰ：coded block pattern）値を決定するように構成され得る。また、いくつかの場合には、エントロピー符号化ユニット５６は係数のランレングスコーディングを実行し得る。さらに、エントロピー符号化ユニット５６、または他の処理ユニットはまた、量子化行列の値など、他のデータをコーディングし得る。

【０１５８】

[0148] 上記で説明されたように、逆量子化ユニット（inverse quantization unit）５８および逆変換処理ユニット６０は、たとえば、参照ブロックとして後で使用するために、ピクセル領域において残差ブロックを再構成するために、それぞれ逆量子化および逆変換を適用する。動き補償ユニット４６は、残差ブロックを参照フレームメモリ６４のフレームのうちの１つの予測ブロックに加算することによって参照ブロックを計算し得る。動き補償ユニット４６はまた、動き推定において使用するためのサブ整数ピクセル値を計算するために、再構成された残差ブロックに１つまたは複数の補間フィルタを適用し得る。加算器６２は、参照フレームメモリ６４に記憶するための再構成されたビデオブロックを生成するために、動き補償ユニット４６によって生成された動き補償予測ブロックに再構成された残差ブロックを加算する。再構成されたビデオブロックは、後続のビデオフレーム中のブロックをインターコーディングするために動き推定ユニット４４および動き補償ユニット４６によって参照ブロックとして使用され得る。

【０１５９】

[0149] 図６は、本開示で説明される技法を実装し得るビデオデコーダ３０の一例を示すブロック図である。この場合も、ビデオデコーダ３０は、例示のためにＨＥＶＣコーディングのコンテキストにおいて説明されるが、他のコーディング規格に関して本開示を限定するものではない。その上、ビデオデコーダ３０は、範囲拡張（range extension）に従って技法を実装するように構成され得る。

【０１６０】

[0150] 図６の例では、ビデオデコーダ３０は、ビデオデータメモリ６９と、エントロピー復号ユニット７０と、予測処理ユニット７１と、逆量子化処理ユニット７６と、逆変換処理ユニット７８と、加算器８０と、参照ピクチャメモリ８２とを含み得る。予測処理ユニット７１は、動き補償ユニット７２とイントラ予測ユニット７４とを含む。ビデオデコーダ３０は、いくつかの例では、図４からのビデオエンコーダ２０に関して説明された符号化パスに概して相反する復号パスを実行し得る。

【０１６１】

[0151] ビデオデータメモリ６９は、ビデオデコーダ３０の構成要素によって復号されるべき、符号化ビデオビットストリーム（encoded video bitstream）などのビデオデータを記憶し得る。ビデオデータメモリ６９に記憶されるビデオデータは、たとえば、ストレージデバイス３４から、カメラなどのローカルビデオソースから、ビデオデータのワイヤードまたはワイヤレスネットワーク通信を介して、あるいは物理データ記憶媒体にアクセスすることによって取得され得る。ビデオデータメモリ６９は、符号化ビデオビットストリームからの符号化ビデオデータを記憶するコード化ピクチャバッファ（ＣＰＢ：coded picture buffer）を形成し得る。

【０１６２】

[0152] 参照ピクチャメモリ８２は、（たとえば、イントラコーディングモードまたは

インターコーディングモードで）ビデオデコーダ３０によってビデオデータを復号する際に使用するための参照ビデオデータを記憶する復号ピクチャバッファ（ＤＰＢ：decoded picture buffer）の一例である。ビデオデータメモリ６９および参照ピクチャメモリ８２は、同期ＤＲＡＭ（ＳＤＲＡＭ）を含むダイナミックランダムアクセスメモリ（ＤＲＡＭ）、磁気抵抗ＲＡＭ（ＭＲＡＭ）、抵抗性ＲＡＭ（ＲＲＡＭ）、または他のタイプのメモリデバイスなど、様々なメモリデバイスのうちのいずれかによって形成され得る。ビデオデータメモリ６９および参照ピクチャメモリ８２は、同じメモリデバイスまたは別個のメモリデバイスによって与えられ得る。様々な例では、ビデオデータメモリ６９は、ビデオデコーダ３０の他の構成要素とともにオンチップであるか、またはそれらの構成要素に対してオフチップであり得る。

【０１６３】

[0153] 復号プロセス中に、ビデオデコーダ３０は、ビデオエンコーダ２０から、符号化ビデオスライスのビデオブロックと、関連するシンタックス要素とを表す符号化ビデオビットストリームを受信する。ビデオデコーダ３０のエントロピー復号ユニット（entropy decoding unit）７０は、量子化された係数と、動きベクトルまたはイントラ予測モードインジケータと、他のシンタックス要素とを生成するために、ビットストリームをエントロピー復号する（entropy decode）。いくつかの例では、エントロピー復号ユニット７０は、エンコーダによって使用されるプロセスとは概して逆であるプロセスを適用し得る。エントロピー復号ユニット７０は、変換係数の１次元アレイを取り出すために、符号化ビットストリームに対してエントロピー復号プロセスを実行する。使用されるエントロピー復号プロセスは、ビデオエンコーダ２０によって使用されたエントロピーコーディング（たとえば、ＣＡＢＡＣ、ＣＡＶＬＣ、ＰＩＰＥ、または上記で説明された他のプロセス）に依存する。本開示で説明される技法によれば、エントロピー復号ユニット７０は、本開示で説明されるように、たとえばＣＡＢＡＣプロセス内で、ＢＡＣプロセスを適用し得る。エンコーダによって使用されたエントロピーコーディングプロセスにおけるウィンドウサイズは、符号化ビットストリーム中でシグナリングされ得るか、または所定のプロセスであり得る。

【０１６４】

[0154] エントロピー復号ユニット７０は、動きベクトルと他のシンタックス要素とを動き補償ユニット７２に転送する。ビデオデコーダ３０は、ビデオスライスレベルおよび／またはビデオブロックレベルでシンタックス要素を受信し得る。

【０１６５】

[0155] 図７は、本開示の技法による、ＣＡＢＡＣを実行するように構成され得る例示的なエントロピー復号ユニット７０のブロック図である。図７のエントロピー復号ユニット７０は、図５で説明されたエントロピー符号化ユニット５６の様式とは逆の様式でＣＡＢＡＣを実行する。ビットストリーム２１８からのコード化ビットがエントロピー復号ユニット７０に入力される。コード化ビットは、それらがバイパスモードを使用してエントロピーコーディングされたのか、通常モードを使用してエントロピーコーディングされたのか否かに基づいて、コンテキストモデラ２２０またはバイパスコーディングエンジン２２２のいずれかに供給される。コード化ビットがバイパスモードでコーディングされた場合、バイパス復号エンジンは、たとえば、バイナリ値シンタックス要素または非バイナリシンタックス要素のビンを取り出すために、ゴロム－ライスまたは指数ゴロム復号を使用することになる。

【０１６６】

[0156] コード化ビットが通常モードでコーディングされた場合、コンテキストモデラ２２０はコード化ビットのための確率モデルを決定し得、通常復号エンジン２２４は、非バイナリ値シンタックス要素のビン（または、バイナリ値の場合、シンタックス要素自体）を生成するためにコード化ビットを復号し得る。コンテキストおよび確率状態σがコンテキストモデラ２２０によって決定された後、通常復号エンジン２２４は、ビン値を復号するためにＢＡＣを実行する。言い換えれば、通常復号エンジン２２４は、コンテキスト

の確率状態を決定し、前にコーディングされたビンと現在の範囲とに基づいてビン値を復号し得る。ビンを復号した後、コンテキストモデラ２２０は、ウィンドウサイズと復号されたビンの値とに基づいてコンテキストの確率状態を更新し得る。

【０１６７】

[0157] 本開示の１つまたは複数の技法によれば、バイナリ算術コーディングプロセスにおいて確率状態を更新するために使用される変数の同じ値（たとえば、ウィンドウサイズ、スケーリングファクタ（α）、および固定確率更新速度のうちの１つまたは複数）を使用することとは対照的に、エントロピー符号化ユニット５６は、異なるコンテキストおよび／または異なるシンタックス要素について変数の異なる値を使用し得る。たとえば、コンテキストモデラ２２０は、複数のコンテキストのうちのコンテキストについて、バイナリ算術コーディングプロセスにおいて確率状態を更新するために使用される変数の値を決定し、決定された値に基づいて確率状態を更新し得る。

【０１６８】

[0158] いくつかの例では、次の確率状態を決定するためにコンテキストモデラ２２０によって使用されるウィンドウサイズはコンテキスト依存にされ得る。たとえば、コンテキストモデラ２２０は、異なるコンテキストについて異なるウィンドウサイズを使用し得る。一例として、コンテキストモデラ２２０は、複数のコンテキストのうちの第１のコンテキストのための第１のウィンドウサイズを決定し、第１のウィンドウサイズとは異なる、複数のコンテキストのうちの第２のコンテキストのための第２のウィンドウサイズを決定し得る。

【０１６９】

[0159] いくつかの例では、上記のコンテキストモデル依存更新方法を、ＪＣＴＶＣ－Ｆ２５４およびＪＣＴＶ－Ｇ７６４におけるものなど、カウンタベース算術コーダに組み込むとき、ウィンドウサイズの値はコンテキストに依存し得る。さらに、各コンテキストは、式（４）からの確率Ｐｉに加えて、ウィンドウサイズにさらに関連し得る。

【０１７０】

[0160] いくつかの例では、コンテキストモデラ２２０は、2^Mに等しいことがあるウィンドウサイズWを使用し得、ここで、Mは正の整数であり得る。したがって、いくつかのコンテキストは同じM値を有し得るが、各コンテキストは、他のコンテキストとは異なり得る、それ自体のM値を有し得る。

【０１７１】

[0161] いくつかの例では、コンテキストモデラ２２０は、ウィンドウサイズのあらかじめ定義されたセットからウィンドウサイズを決定し得る。たとえば、可能なM値のセットがあらかじめ定義され得、たとえば、Mは、両端値を含む、４から７までにわたることがある。いくつかの例では、エントロピー復号ユニット７０は、スライスヘッダ、あるいはピクチャパラメータセット、アクティブパラメータセット、シーケンスパラメータセット、またはビデオパラメータセットを含むパラメータセットから、可能なウィンドウサイズのセットの指示（たとえば、可能なM値のセットの指示）を復号し得る。

【０１７２】

[0162] いくつかの例では、各コンテキストに関連するウィンドウサイズ（たとえば、Mの値）はあらかじめ定義され得る。いくつかの例では、ウィンドウサイズは、さらに、スライスタイプおよび／または（たとえば、ＨＥＶＣではｔｅｍｐｏｒａｌＩｄと呼ばれる）時間識別子に依存し得る。いくつかの例では、ウィンドウサイズは、さらに、ピクチャタイプ（またはＮＡＬユニットタイプ）、たとえば、ピクチャがランダムアクセスピクチャであるか否かに依存し得る。

【０１７３】

[0163] いくつかの例では、エントロピー復号ユニット７０は、スライスヘッダ／ピクチャパラメータセット／アクティブパラメータセット／シーケンスパラメータセット中でなど、ビットストリームから、各コンテキストに関連するウィンドウサイズ（たとえば、Mの値）を復号し得る。たとえば、各コンテキストのためのデフォルトウィンドウサイズ

が最初にあらかじめ定義され得る。各それぞれのコンテキストについて、エントロピー復号ユニット７０は、デフォルトウィンドウサイズがそれぞれのコンテキストのために使用されるかどうかを示す、それぞれのシンタックス要素（たとえば、フラグ）を復号し得る。デフォルトウィンドウサイズがそれぞれのコンテキストのために使用されない場合、エントロピー復号ユニット７０は、デフォルトウィンドウサイズに基づいて、実際の使用されるウィンドウサイズを差分復号し得る。いくつかの例では、すべてのコンテキストの（すなわち、デフォルトウィンドウサイズが使用されるかどうかを示す）シンタックス要素が一緒に編成され得、エントロピー復号ユニット７０は、これらのシンタックス要素を復号するためにランレングスコーディングを利用し得る。いくつかの例では、コンテキストモデラ２２０は、実際に使用されるウィンドウサイズとデフォルトウィンドウサイズとの間の差分をコーディングするとき、マッピングテーブルを利用し得る。たとえば、デフォルトのＭ値が６に等しい場合、可能なＭ値は、４、５、６、および７である。マッピングテーブルは次のように定義され得る。

【０１７４】

【表７】

実際のM値	4	5	6	7
コーディングされるべき値	0	1	-	2

【０１７５】

[0164] いくつかの例では、エントロピー復号ユニット７０は、各コンテキストについて実際のウィンドウサイズとデフォルトウィンドウサイズとの間の差分を直接復号し得る。たとえば、デフォルトＭ値が４である場合、エントロピー復号ユニット７０は、各コンテキストについてＭ－４の値を復号し得る。

【０１７６】

[0165] いくつかの例では、エントロピー復号ユニット７０は、現在スライス中のコンテキストのためのすべてのウィンドウサイズが、前にコーディングされたスライス中の対応するコンテキストのための継承されたウィンドウサイズである（すなわち、前にコーディングされたスライス中の対応するコンテキストのためのウィンドウサイズに等しく設定される）かどうかを示す、第１のシンタックス要素を復号し得る。一例では、「前に復号されたスライス」は、現在スライスと同じスライスタイプ、または同じスライスタイプと量子化パラメータの両方、または同じスライスタイプと時間レイヤの両方、および／あるいは同じ初期化された量子化パラメータを有する、前にコーディングされたスライスとして定義され得る。いくつかの例では、前のスライスは、ＤＰＢ中に存在するピクチャに属することが必要であり得、参照ピクチャとして現在ピクチャのために使用され得、特に、ＨＥＶＣベースプラットフォームの場合のように、前のスライスは、参照ピクチャセット（ＲＰＳ）中のピクチャ、さらには、ＲＰＳの以下のサブセット、すなわち、ＲｅｆＰｉｃＳｅｔＳｔＣｕｒｒＢｅｆｏｒｅ、ＲｅｆＰｉｃＳｅｔＳｔＣｕｒｒＡｆｔｅｒ、およびＲｅｆＰｉｃＳｅｔＬｔＣｕｒｒのうちの１つ中のピクチャに属することが必要とされ得る。

【０１７７】

[0166] いくつかの例では、エントロピー復号ユニット７０は、デフォルトウィンドウサイズが（たとえば、現在スライス中の）複数のコンテキストのために使用されるかどうかを示す第１のシンタックス要素を復号し得る。デフォルトウィンドウサイズが複数のコンテキストのために使用されない場合、エントロピー復号ユニット７０は、コンテキストのためのウィンドウサイズを示す第２のシンタックス要素を復号し得る。たとえば、エントロピー復号ユニット７０は、コンテキストのためのウィンドウサイズとデフォルトウィンドウサイズとの間の差分を示す第２のシンタックス要素を復号し得る。

【０１７８】

[0167] 別の例では、エントロピー復号ユニット７０は、たとえば、前のスライスまたはピクチャからのコード化情報に基づいて、ウィンドウサイズを導出し得る。たとえば、

エントロピー復号ユニット７０は、１つのコンテキストが追跡されるに関連する前のスライス中の復号されたビンを追跡し得る。可能なウィンドウサイズの各候補について、エントロピー復号ユニット７０は、これらのビンをコーディングするために消費されるビットを取得し得る。エントロピー復号ユニット７０は、これらのビンをコーディングするための最小ビットを生じるウィンドウサイズを、このコンテキストのためのウィンドウサイズとして選択し得る。エントロピー復号ユニット７０は、後続のスライス／ピクチャを復号するために、選択されたウィンドウサイズを使用し得る。

【０１７９】

[0168] いくつかの例では、算術コーダにおいて次の確率状態または確率更新速度を決定するために使用される「ウィンドウサイズ」はシンタックス要素固有であり得る。たとえば、シンタックス要素のビンを符号化するためにコンテキストを使用するとき、コンテキストモデラ２２０は、シンタックス要素タイプに基づいてコンテキストのためのウィンドウサイズを決定し得る。一例として、コーディングユニットスプリットシンタックス要素のビンと、コーディングユニットスキップフラグシンタックス要素のビンとをコーディングするときの、コンテキストを更新するために使用されるウィンドウサイズは同じ、たとえば、１６（すなわち、Ｍ＝４）であり得る。

【０１８０】

[0169] いくつかの例では、上記の（１つまたは複数の）技法は特定のコンテキストに適用可能であり得る。すなわち、コンテキストのサブセットは、デフォルトウィンドウサイズではなく更新された「ウィンドウサイズ」を使用し得る。いくつかの例では、上記の（１つまたは複数の）技法は特定のスライスタイプに適用可能であり得る。

【０１８１】

[0170] ビンが通常復号エンジン２２４によって復号された後、逆方向バイナライザ（reverse binarizer）２３０は、ビンを非バイナリ値シンタックス要素の値に変換するために逆方向マッピング（reverse mapping）を実行し得る。

【０１８２】

[0171] 図６に戻ると、いくつかの例では、エントロピー復号ユニット７０（または逆量子化ユニット７６）は、ビデオエンコーダ２０のエントロピー符号化ユニット５６（または量子化ユニット５４）によって使用された走査モードをミラーリングする走査を使用して受信値を走査し得る。係数の走査は逆量子化ユニット７６において実行され得るが、走査は、例示のために、エントロピー復号ユニット７０によって実行されるものとして説明される。さらに、説明しやすいように別個の機能ユニットとして示されているが、エントロピー復号ユニット７０、逆量子化ユニット７６、およびビデオデコーダ３０の他のユニットの構造および機能は互いに高度に統合され得る。

【０１８３】

[0172] 逆量子化ユニット７６は、ビットストリーム中で与えられ、エントロピー復号ユニット７０によって復号された、量子化された変換係数を逆量子化、すなわち、量子化解除する。逆量子化プロセスは、たとえば、ＨＥＶＣのまたはH．２６４復号規格によって定義されたいくつかの例と同様の、従来のプロセスを含み得る。逆量子化プロセスは、量子化の程度を決定し、同様に、適用されるべき逆量子化の程度を決定するための、ＣＵについてビデオエンコーダ２０によって計算される量子化パラメータ（quantization parameter）ＱＰの使用を含み得る。逆量子化ユニット７６は、係数が１次元アレイから２次元アレイに変換される前または変換された後に変換係数を逆量子化し得る。

【０１８４】

[0173] 逆変換処理ユニット７８は、逆量子化された変換係数に逆変換を適用する。いくつかの例では、逆変換処理ユニット７８は、ビデオエンコーダ２０からのシグナリングに基づいて、あるいはブロックサイズ、コーディングモードなどの１つまたは複数のコーディング特性から変換を推論することによって、逆変換を決定し得る。いくつかの例では、逆変換処理ユニット７８は、現在ブロックを含むＬＣＵのための４分木のルートノードにおけるシグナリングされた変換に基づいて、現在ブロックに適用すべき変換を決定し得

る。代替的に、変換は、ＬＣＵ４分木中のリーフノードＣＵのためのＴＵ４分木のルートにおいてシグナリングされ得る。いくつかの例では、逆変換処理ユニット７８は、逆変換処理ユニット７８が、復号されている現在ブロックの変換係数に２つまたはそれ以上の逆変換を適用する、カスケード逆変換（cascaded inverse transform）を適用し得る。

【０１８５】

[0174] さらに、逆変換処理ユニットは、本開示の上記で説明された技法に従って、変換ユニットパーティションを生成するために逆変換を適用し得る。

【０１８６】

[0175] イントラ予測処理ユニット７４は、シグナリングされたイントラ予測モードと、現在フレームの、前に復号されたブロックからのデータとに基づいて、現在フレームの現在ブロックのための予測データを生成し得る。取り出された動き予測方向と、基準フレームインデックスと、計算された現在動きベクトル（たとえば、マージモードに従って隣接ブロックからコピーされた動きベクトル）とに基づいて、動き補償ユニットは現在部分のための動き補償ブロックを生成する。これらの動き補償ブロックは、本質的に、残差データを生成するために使用される予測ブロックを再現する。

【０１８７】

[0176] 動き補償ユニット７２は、場合によっては、補間フィルタに基づく補間を実行して、動き補償ブロックを生成し得る。サブピクセル精度をもつ動き推定のために使用されるべき補間フィルタのための識別子が、シンタックス要素中に含まれ得る。動き補償ユニット７２は、参照ブロックのサブ整数ピクセルのための補間値を計算するために、ビデオブロックの符号化中にビデオエンコーダ２０によって使用された補間フィルタを使用し得る。動き補償ユニット７２は、受信されたシンタックス情報に従って、ビデオエンコーダ２０によって使用された補間フィルタを決定し、予測ブロックを生成するためにその補間フィルタを使用し得る。

【０１８８】

[0177] さらに、動き補償ユニット７２およびイントラ予測処理ユニット７４は、ＨＥＶＣの例では、符号化ビデオシーケンスの（１つまたは複数の）フレームを符号化するために使用されたＬＣＵのサイズを決定するために、（たとえば、４分木によって与えられる）シンタックス情報の一部を使用し得る。動き補償ユニット７２およびイントラ予測処理ユニット７４はまた、符号化ビデオシーケンスのフレームの各ＣＵがどのように分割されるか（および、同様に、サブＣＵがどのように分割されるか）を記述する分割情報を決定するために、シンタックス情報を使用し得る。シンタックス情報はまた、各分割がどのように符号化されるかを示すモード（たとえば、イントラ予測またはインター予測、およびイントラ予測の場合はイントラ予測符号化モード）と、各インター符号化ＰＵについての１つまたは複数の参照フレーム（および／またはそれらの参照フレームの識別子を含んでいる参照リスト）と、符号化ビデオシーケンスを復号するための他の情報とを含み得る。

【０１８９】

[0178] 加算器８０は、復号ブロックを形成するために、残差ブロックを、動き補償ユニット７２またはイントラ予測処理ユニット７４によって生成される対応する予測ブロックと合成する。所望される場合、ブロッキネスアーティファクトを除去するために、復号ブロックをフィルタ処理するためにデブロッキングフィルタも適用され得る。復号ビデオブロックは、次いで、参照ピクチャメモリ８２に記憶され、参照ピクチャメモリ８２は、参照ブロックを後続の動き補償に与え、また、（図１のディスプレイデバイス３１などの）ディスプレイデバイス上での提示のために復号ビデオを生成する。

【０１９０】

[0179] 図８は、通常コーディングモードを使用する所与のビン値ｂｉｎＶａｌのためのバイナリ算術符号化プロセスを示す。算術符号化エンジンの内部状態は、通常通り、２つの量、すなわち、現在の間隔範囲Ｒおよび現在のコード間隔のベース（下側端点）Ｌによって特徴づけられる。しかしながら、（通常モードとバイパスモードの両方において）

これらのレジスタをＣＡＢＡＣエンジンに記憶するために必要とされる精度は、それぞれ９ビットおよび１０ビットまで低減され得ることに留意されたい。確率状態インデックスδをもつコンテキストにおいて観測される所与のバイナリ値ｂｉｎＶａｌおよびＭＰＳ（δ％２）の値の符号化は、以下のように４つの基本ステップのシーケンスにおいて実行される。

【０１９１】

[0180] 第１の主要なステップにおいて、現在の間隔が、所与の確率推定値に従って再分割される。この間隔再分割プロセスは、図８中の流れ図の最上のボックスに示されているように３つの基本演算を伴う。最初に、現在の間隔範囲Ｒが、４つのセルへの全範囲$2^8 \leqq R \leqq 2^9$の等区分を使用して、量子化された値Ｑ（Ｒ）によって近似される。しかし、ＣＡＢＡＣエンジンにおいて対応する代表的な量子化された範囲値Q_0、Q_1、Q_2、およびQ_3を明示的に使用する代わりに、すなわち、以下の式（１４）に従って、シフト演算とビットマスキング演算との組合せによって効率的に計算され得る、それの量子化器インデックスρのみによって対処される。

【０１９２】

【数１６】

$$\rho = (R \gg 6)\ \&\ 3 \qquad (14)$$

【０１９３】

[0181] 次いで、このインデックスρおよび確率状態インデックスδが、図８に示されているように、（近似）ＬＰＳ関係のサブ間隔範囲R_{LPS}を決定するために、２ＤテーブルＴａｂＲａｎｇｅＬＰＳ中のエントリとして使用される。ここで、テーブルＴａｂＲａｎｇｅＬＰＳは、８ビット精度での、０≦（δ＞＞１）≦６３および０≦ρ≦３である場合の、$p_\sigma \cdot Q_\rho$のためのすべての６４×４個のあらかじめ計算された積値（pre-computed product value）を含んでいる。

【０１９４】

[0182] ＭＰＳのためのデュアルサブ間隔範囲が与えられれば、所与のビン値ｂｉｎＶａｌに対応するサブ間隔が、符号化プロセスの第２のステップにおいて選定される。ｂｉｎＶａｌがＭＰＳ値に等しい場合、Ｌが不変であるように、下側サブ間隔が選定され（図８中の分岐の右経路）、そうでない場合、R_{LPS}に等しい範囲をもつ上側サブ間隔が選択される（図８中の左分岐）。通常算術符号化プロセスの第３のステップにおいて、確率状態の更新が、（たとえば、式（２）を使用して）上記で説明されたように実行され（図８中のグレーの影付きボックス）、最終的に、第４のステップは、Ｍａｒｐｅによって説明されたようにレジスタＬおよびＲの再正規化（図８中の「ＲｅｎｏｒｍＥ」ボックス）からなる。

【０１９５】

[0183] ２ＤテーブルＴａｂＲａｎｇｅＬＰＳは以下のように定義され得る。

【０１９６】

【数１７】

```
TabRangeLPS[64][4] =
{
 { 128, 176, 208, 240},
 { 128, 167, 197, 227},
 { 128, 158, 187, 216},
 { 123, 150, 178, 205},
```

{ 116, 142, 169, 195},
{ 111, 135, 160, 185},
{ 105, 128, 152, 175},
{ 100, 122, 144, 166},
{ 95, 116, 137, 158},
{ 90, 110, 130, 150},
{ 85, 104, 123, 142},
{ 81, 99, 117, 135},
{ 77, 94, 111, 128},
{ 73, 89, 105, 122},
{ 69, 85, 100, 116},
{ 66, 80, 95, 110},
{ 62, 76, 90, 104},
{ 59, 72, 86, 99},
{ 56, 69, 81, 94},
{ 53, 65, 77, 89},
{ 51, 62, 73, 85},
{ 48, 59, 69, 80},
{ 46, 56, 66, 76},
{ 43, 53, 63, 72},
{ 41, 50, 59, 69},
{ 39, 48, 56, 65},
{ 37, 45, 54, 62},
{ 35, 43, 51, 59},
{ 33, 41, 48, 56},
{ 32, 39, 46, 53},
{ 30, 37, 43, 50},
{ 29, 35, 41, 48},
{ 27, 33, 39, 45},
{ 26, 31, 37, 43},
{ 24, 30, 35, 41},
{ 23, 28, 33, 39},
{ 22, 27, 32, 37},
{ 21, 26, 30, 35},
{ 20, 24, 29, 33},
{ 19, 23, 27, 31},
{ 18, 22, 26, 30},
{ 17, 21, 25, 28},
{ 16, 20, 23, 27},
{ 15, 19, 22, 25},
{ 14, 18, 21, 24},
{ 14, 17, 20, 23},
{ 13, 16, 19, 22},
{ 12, 15, 18, 21},
{ 12, 14, 17, 20},
{ 11, 14, 16, 19},
{ 11, 13, 15, 18},
{ 10, 12, 15, 17},
{ 10, 12, 14, 16},
{ 9, 11, 13, 15},

```
    {  9, 11, 12, 14},
    {  8, 10, 12, 14},
    {  8,  9, 11, 13},
    {  7,  9, 11, 12},
    {  7,  9, 10, 12},
    {  7,  8, 10, 11},
    {  6,  8,  9, 11},
    {  6,  7,  9, 10},
    {  6,  7,  8,  9},
    {  2,  2,  2,  2}
};
```

【０１９７】

[0184] 例示的なＣＡＢＡＣ復号プロセスは、ＨＥＶＣ規格のセクション９．３．４．３．２．２において見つけられ得る。

【０１９８】

[0185] 図９は、残差４分木に基づく変換方式を示す概念図である。残差ブロックの様々な特性を適応させるために、ＨＥＶＣでは残差４分木（ＲＱＴ）を使用する変換コーディング構造が適用され、これは、
http://www.hhi.fraunhofer.de/departments/video-coding-analytics/research-groups/image-video-coding/hevc-high-efficiency-video-coding/transform-coding-using-the-residual-quadtree-rqt.html
に手短に記載されている。

【０１９９】

[0186] 各ピクチャが、特定のタイルまたはスライスについてラスタ走査順序でコーディングされるコーディングツリーユニット（ＣＴＵ：coding tree unit）に分割される。ＣＴＵは、正方形ブロックであり、４分木、すなわち、コーディングツリーのルートを表す。ＣＴＵサイズは８×８から６４×６４ルーマサンプルにわたり得るが、一般に６４×６４が使用される。各ＣＴＵは、さらに、コーディングユニット（ＣＵ）と呼ばれるより小さい正方形ブロックにスプリットされ得る。ＣＴＵがＣＵに再帰的にスプリットされた後、各ＣＵはＰＵおよびＴＵにさらに分割される。ＴＵへのＣＵの区分は、４分木手法に基づいて再帰的に行われ、したがって、各ＣＵの残差信号は、ツリー構造、すなわち、残差４分木（ＲＱＴ）によってコーディングされる。ＲＱＴは、４×４から３２×３２ルーマサンプルまでのＴＵサイズを可能にする。図９は、ＣＵが、文字ａ～ｊで標示された１０個のＴＵを含む一例と、対応するブロック区分とを示す。ＲＱＴの各ノードは、実際は変換ユニット（ＴＵ）である。個々のＴＵは、深度優先トラバーサル（depth-first traversal）による再帰的Ｚ走査に従う、アルファベット順として図に示された深度優先ツリートラバーサル順序で処理される。４分木手法は、残差信号の変動する空間周波数特性に対する変換の適応を可能にする。一般に、より大きい空間サポートを有するより大きい変換ブロックサイズは、より良い周波数解像度を与える。しかしながら、より小さい空間サポートを有するより小さい変換ブロックサイズは、より良い空間解像度を与える。その２つ、すなわち、空間解像度と周波数解像度との間のトレードオフは、たとえばレートひずみ最適化技法に基づいて、エンコーダモード決定によって選定される。レートひずみ最適化技法は、各コーディングモード（たとえば、特定のＲＱＴスプリッティング構造）についてコーディングビットと再構成ひずみとの加重和、すなわち、レートひずみコストを計算し、最小レートひずみコストをもつコーディングモードを最良のモードとして選択する。

【０２００】

[0187] ３つのパラメータ、すなわち、ツリーの最大深度（maximum depth of the tree）、最小許容変換サイズ（minimum allowed transform size）および最大許容変換サイズ（maximum allowed transform size）がＲＱＴにおいて定義される。ＨＥＶＣのいくつかの例では、最小および最大変換サイズは、前の段落で述べられたサポートされるブロッ

ク変換に対応する、４×４から３２×３２サンプルまでの範囲内で変動することがある。ＲＱＴの最大許容深度はＴＵの数を制限する。０に等しい最大深度は、各含まれたＴＵが最大許容変換サイズ、たとえば、３２×３２に達した場合、ＣＴＵがこれ以上スプリットされ得ないことを意味する。

【０２０１】

[0188] すべてのこれらのパラメータは、相互作用し、ＲＱＴ構造に影響を及ぼす。ルートＣＴＵサイズが６４×６４であり、最大深度が０に等しく、最大変換サイズが３２×３２に等しい場合について考える。この場合、ＣＴＵは、さもなければ、それが、許容されない６４×６４ＴＵにつながることになるので、少なくとも１回区分されなければならない。ＲＱＴパラメータ、すなわち、最大ＲＱＴ深度、最小および最大変換サイズは、シーケンスパラメータセットレベルにおいてビットストリーム中で送信される。ＲＱＴ深度に関して、イントラコード化ＣＵとインターコード化ＣＵとについて異なる値が指定され、シグナリングされ得る。

【０２０２】

[0189] ４分木変換は、イントラ残差ブロックとインター残差ブロックの両方のために適用される。一般に、現在の残差４分木パーティションの同じサイズのＤＣＴ－ＩＩ変換が残差ブロックのために適用される。しかしながら、現在の残差４分木ブロックが４×４であり、イントラ予測によって生成される場合、上記の４×４ＤＳＴ－ＶＩＩ変換が適用される。

【０２０３】

[0190] ＨＥＶＣでは、より大きいサイズの変換、たとえば、６４×６４変換は、主に、および比較的より小さい解像度のビデオに対する比較的高い複雑さを考慮して、それらの限られた利益により、採用されない。

【０２０４】

[0191] 図１０は、係数グループに基づく例示的な係数走査を示す概念図である。ＴＵサイズかかわらず、変換ユニットの残差は、各々がＴＵの４×４ブロックの係数を含んでいる非重複係数グループ（ＣＧ：coefficient group）でコーディングされる。たとえば、３２×３２ＴＵは全体として６４個のＣＧを有し、１６×１６ＴＵは全体として１６個のＣＧを有する。ＴＵ内のＣＧは、あるあらかじめ定義された走査順序に従ってコーディングされ得る。各ＣＧをコーディングするとき、現在ＣＧ内の係数は、４×４ブロックについて、あるあらかじめ定義された走査順序に従って走査され、コーディングされる。図１０は、４つのＣＧを含んでいる８×８ＴＵについての係数走査を示す。

【０２０５】

[0192] シンタックス要素テーブルは以下のように定義される。

【０２０６】

【表８】

7.3.8.11　残差コーディングシンタックス

residual_coding(x0, y0, log2TrafoSize, cIdx) {	記述子
if(transform_skip_enabled_flag && !cu_transquant_bypass_flag && (log2TrafoSize == 2))	
transform_skip_flag[x0][y0][cIdx]	ae(v)
last_sig_coeff_x_prefix	ae(v)
last_sig_coeff_y_prefix	ae(v)
if(last_sig_coeff_x_prefix > 3)	
last_sig_coeff_x_suffix	ae(v)
if(last_sig_coeff_y_prefix > 3)	
last_sig_coeff_y_suffix	ae(v)
lastScanPos = 16	
lastSubBlock = (1 << (log2TrafoSize − 2)) * (1 << (log2TrafoSize − 2)) − 1	
do {	
if(lastScanPos == 0) {	
lastScanPos = 16	
lastSubBlock− −	
}	
lastScanPos− −	
xS = ScanOrder[log2TrafoSize − 2][scanIdx][lastSubBlock][0]	
yS = ScanOrder[log2TrafoSize − 2][scanIdx][lastSubBlock][1]	
xC = (xS << 2) + ScanOrder[2][scanIdx][lastScanPos][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][lastScanPos][1]	
} while((xC != LastSignificantCoeffX) || (yC != LastSignificantCoeffY))	
for(i = lastSubBlock; i >= 0; i− −) {	
xS = ScanOrder[log2TrafoSize − 2][scanIdx][i][0]	
yS = ScanOrder[log2TrafoSize − 2][scanIdx][i][1]	
inferSbDcSigCoeffFlag = 0	
if((i < lastSubBlock) && (i > 0)) {	
coded_sub_block_flag[xS][yS]	ae(v)
inferSbDcSigCoeffFlag = 1	
}	
for(n = (i == lastSubBlock) ? lastScanPos − 1 : 15; n >= 0; n− −) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	

if(coded_sub_block_flag[xS][yS] && (n > 0 || !inferSbDcSigCoeffFlag)) {	
sig_coeff_flag[xC][yC]	ae(v)
if(sig_coeff_flag[xC][yC])	
inferSbDcSigCoeffFlag = 0	
}	
}	
firstSigScanPos = 16	
lastSigScanPos = −1	
numGreater1Flag = 0	
lastGreater1ScanPos = −1	
for(n = 15; n >= 0; n− −) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
if(sig_coeff_flag[xC][yC]) {	
if(numGreater1Flag < 8) {	
coeff_abs_level_greater1_flag[n]	ae(v)
numGreater1Flag++	
if(coeff_abs_level_greater1_flag[n] && lastGreater1ScanPos == −1)	
lastGreater1ScanPos = n	
}	
if(lastSigScanPos == −1)	
lastSigScanPos = n	
firstSigScanPos = n	
}	
}	
signHidden = (lastSigScanPos − firstSigScanPos > 3 && !cu_transquant_bypass_flag)	
if(lastGreater1ScanPos != −1)	
coeff_abs_level_greater2_flag[lastGreater1ScanPos]	ae(v)
for(n = 15; n >= 0; n− −) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
if(sig_coeff_flag[xC][yC] && (!sign_data_hiding_enabled_flag || !signHidden || (n != firstSigScanPos)))	
coeff_sign_flag[n]	ae(v)
}	
numSigCoeff = 0	
sumAbsLevel = 0	
for(n = 15; n >= 0; n− −) {	

xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
if(sig_coeff_flag[xC][yC]) {	
baseLevel = 1 + coeff_abs_level_greater1_flag[n] + coeff_abs_level_greater2_flag[n]	
if(baseLevel == ((numSigCoeff < 8) ? ((n == lastGreater1ScanPos) ? 3 : 2) : 1))	
coeff_abs_level_remaining[n]	ae(v)
TransCoeffLevel[x0][y0][cIdx][xC][yC] = (coeff_abs_level_remaining[n] + baseLevel) * (1 − 2 * coeff_sign_flag[n])	
if(sign_data_hiding_enabled_flag && signHidden) {	
sumAbsLevel += (coeff_abs_level_remaining[n] + baseLevel)	
if((n == firstSigScanPos) && ((sumAbsLevel % 2) == 1))	
TransCoeffLevel[x0][y0][cIdx][xC][yC] = −TransCoeffLevel[x0][y0][cIdx][xC][yC]	
}	
numSigCoeff++	
}	
}	
}	
}	

【０２０７】

[0193] 各色成分について、現在ＴＵが少なくとも１つの非０係数を有するかどうかを示すために、１つのフラグが最初にシグナリングされ得る。少なくとも１つの非０係数がある場合、ＴＵ中の係数走査順序（coefficient scan order）における最後有意係数（last significant coefficient）の位置が、変換ユニットの左上隅に対する座標を用いて明示的にコーディングされる。座標の垂直または水平成分は、それのプレフィックスおよびサフィックスによって表され、ここにおいて、プレフィックスは短縮ライス（ＴＲ：truncated rice）を用いて２値化され、サフィックスは固定長を用いて２値化される。

【０２０８】

[0194] セマンティクス：

[0195] ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｘ＿ｐｒｅｆｉｘは、変換ブロック内の走査順序における最後有意係数の列位置のプレフィックスを指定する。ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｘ＿ｐｒｅｆｉｘの値は、両端値を含む、０～（ｌｏｇ２ＴｒａｆｏＳｉｚｅ＜＜１）－１の範囲内にあるものとする。

【０２０９】

[0196] ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｙ＿ｐｒｅｆｉｘは、変換ブロック内の走査順序における最後有意係数の行位置のプレフィックスを指定する。ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｙ＿ｐｒｅｆｉｘの値は、両端値を含む、０～（ｌｏｇ２ＴｒａｆｏＳｉｚｅ＜＜１）－１の範囲内にあるものとする。

【０２１０】

[0197] ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｘ＿ｓｕｆｆｉｘは、変換ブロック内の走査順序における最後有意係数の列位置のサフィックスを指定する。ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｘ＿ｓｕｆｆｉｘの値は、両端値を含む、０～（１＜＜（（ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｘ＿ｐｒｅｆｉｘ＞＞１）－１））－１の範囲内にあるものとする。

【０２１１】

[0198] 変換ブロックＬａｓｔＳｉｇｎｉｆｉｃａｎｔＣｏｅｆｆＸ内の走査順序における最後有意係数の列位置は、以下のように導出される。

－　ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｘ＿ｓｕｆｆｉｘが存在しない場合、以下が適用される。

【０２１２】

【数１８】

$$LastSignificantCoeffX = last\_sig\_coeff\_x\_prefix$$

【０２１３】

－　そうでない場合（ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｘ＿ｓｕｆｆｉｘが存在する）、以下が適用される。

【０２１４】

【数１９】

$$LastSignificantCoeffX = (1 << ((last\_sig\_coeff\_x\_prefix >> 1) - 1)) * (2 + (last\_sig\_coeff\_x\_prefix \& 1)) + last\_sig\_coeff\_x\_suffix$$

【０２１５】

[0199]　ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｙ＿ｓｕｆｆｉｘは、変換ブロック内の走査順序における最後有意係数の行位置のサフィックスを指定する。ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｙ＿ｓｕｆｆｉｘの値は、両端値を含む、０～（１＜＜（（ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｙ＿ｐｒｅｆｉｘ＞＞１）－１））－１の範囲内にあるものとする。

【０２１６】

[0200]　変換ブロックＬａｓｔＳｉｇｎｉｆｉｃａｎｔＣｏｅｆｆＹ内の走査順序における最後有意係数の行位置は、以下のように導出される。

－　ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｙ＿ｓｕｆｆｉｘが存在しない場合、以下が適用される。

【０２１７】

【数２０】

$$LastSignificantCoeffY = last\_sig\_coeff\_y\_prefix$$

【０２１８】

－　そうでない場合（ｌａｓｔ＿ｓｉｇ＿ｃｏｅｆｆ＿ｙ＿ｓｕｆｆｉｘが存在する）、以下が適用される。

【０２１９】

【数２１】

$$LastSignificantCoeffY = (1 << ((last\_sig\_coeff\_y\_prefix >> 1) - 1)) * (2 + (last\_sig\_coeff\_y\_prefix \& 1)) + last\_sig\_coeff\_y\_suffix$$

【０２２０】

[0201]　ｓｃａｎＩｄｘが２に等しいとき、座標は以下のようにスワップされる。

【０２２１】

【数２２】

$$(LastSignificantCoeffX, LastSignificantCoeffY)=Swap(LastSignificantCoeffX, LastSignificantCoeffY)$$

【０２２２】

[0202]　コーディングされたそのような位置、またＣＧの係数走査順序とともに、（走査順序における）最後ＣＧを除くＣＧのために、それが非０係数を含んでいるかどうかを示す１つのフラグがさらにシグナリングされる。

【０２２３】

[0203]　ＣＧフラグのコンテキストモデリング。１つのＣＧが非０係数を有するかどうか、すなわち、ＣＧフラグ（ＨＥＶＣ仕様におけるｃｏｄｅｄ＿ｓｕｂ＿ｂｌｏｃｋ＿ｆｌａｇ）をコーディングするとき、隣接ＣＧの情報が、コンテキストを構築するために利用される。より具体的に言えば、ＣＧフラグをコーディングするためのコンテキスト選択は次のように定義される。

【０２２４】

【数２３】

(右CG利用可能＆＆右CGのフラグが1に等しい)||(下CG利用可能＆＆下CGのフラグが1に等しい)

【０２２５】

[0204] ここで、右および下ＣＧは、現在ＣＧに近接する２つの隣接ＣＧである。たとえば、図１０では、左上４×４ブロックをコーディングするとき、右ＣＧは右上４×４ブロックとして定義され、下ＣＧは左下４×４ブロックとして定義される。

【０２２６】

[0205] クロマおよびルーマは、コンテキストの異なるセットを使用するが、それらのうちの１つを選択するための同じルールを用いることに留意されたい。

【０２２７】

[0206] コンテキストインデックス増分の導出の詳細は、ＨＥＶＣの９．３．４．２．４において見つけられ得る。

【０２２８】

[0207] １つのＣＧ内での変換係数コーディング。非０係数を含んでいることがあるＣＧについて、有意フラグ（ｓｉｇｎｉｆｉｃａｎｔ＿ｆｌａｇ）、（ｃｏｅｆｆ＿ａｂｓ＿ｌｅｖｅｌ＿ｇｒｅａｔｅｒ１＿ｆｌａｇ、ｃｏｅｆｆ＿ａｂｓ＿ｌｅｖｅｌ＿ｇｒｅａｔｅｒ２＿ｆｌａｇおよびｃｏｅｆｆ＿ａｂｓ＿ｌｅｖｅｌ＿ｒｅｍａｉｎｉｎｇを含む）係数の絶対値および符号情報（ｃｏｅｆｆ＿ｓｉｇｎ＿ｆｌａｇ）が、あらかじめ定義された４×４係数走査順序に従って各係数についてさらにコーディングされ得る。変換係数レベルのコーディングは複数の走査パスに分離される。

【０２２９】

[0208] １）第１のビンコーディングの第１のパス。このパスにおいて、特定の変換係数が０に等しいことが導出され得ることを除いて、１つのＣＧ内の各位置における変換係数のすべての第１のビン（またはビンインデックス０、ｂｉｎ０）がコーディングされる。

【０２３０】

[0209] 変数ｓｉｇＣｔｘは、現在ＴＵの左上ｐｏｓｔｉｏｎに対する現在ロケーションと、色成分インデックスｃＩｄｘと、変換ブロックサイズと、シンタックス要素ｃｏｄｅｄ＿ｓｕｂ＿ｂｌｏｃｋ＿ｆｌａｇの前に復号されたビンとに依存する。異なるルールがＴＵサイズに応じて適用される。コンテキストインデックス増分の選択の例示的な詳細は、ＨＥＶＣの９．３．４．２．５において定義されている。

【０２３１】

[0210] ２）第２のビンコーディングの第２のパス。ｃｏｅｆｆ＿ａｂｓ＿ｌｅｖｅｌ＿ｇｒｅａｔｅｒ１＿ｆｌａｇのコーディングがこのパスにおいて適用される。コンテキストモデリングは、色成分インデックスと、現在サブブロック走査インデックスと、現在サブブロック内の現在係数走査インデックスとに依存する。コンテキストインデックス増分の選択の例示的な詳細は、ＨＥＶＣの９．３．４．２．６において定義されている。

【０２３２】

[0211] ３）第３のビンコーディングの第３のパス。ｃｏｅｆｆ＿ａｂｓ＿ｌｅｖｅｌ＿ｇｒｅａｔｅｒ２＿ｆｌａｇのコーディングがこのパスにおいて適用される。コンテキストモデリングは、ｃｏｅｆｆ＿ａｂｓ＿ｌｅｖｅｌ＿ｇｒｅａｔｅｒ１＿ｆｌａｇによって使用されるものと同様である。コンテキストインデックス増分の選択の例示的な詳細は、ＨＥＶＣの９．３．４．２．７において定義されている。

【０２３３】

[0212] スループットを改善するために、第２および第３のパスはＣＧ中のすべての係数を処理するとは限らないことに留意されたい。ＣＧ中の最初の８つのｃｏｅｆｆ＿ａｂｓ＿ｌｅｖｅｌ＿ｇｒｅａｔｅｒ１＿ｆｌａｇが通常モードでコーディングされる。その後、値は、シンタックスｃｏｅｆｆ＿ａｂｓ＿ｌｅｖｅｌ＿ｒｅｍａｉｎｉｎｇによって第５のパスにおいてバイパスモードでコーディングされるために残される。同様に、１よりも大きい値をもつＣＧ中の第１の係数のためのｃｏｅｆｆ＿ａｂｓ＿ｌｅｖｅｌ＿ｇｒ

ｅａｔｅｒ２＿ｆｌａｇのみがコーディングされる。ＣＧの１よりも大きい値をもつ係数の残りは、値をコーディングするためにｃｏｅｆｆ＿ａｂｓ＿ｌｅｖｅｌ＿ｒｅｍａｉｎｉｎｇを使用する。この方法は、係数レベルのための通常ビンの数を、ＣＧごとに最大９つ、すなわち、ｃｏｅｆｆ＿ａｂｓ＿ｌｅｖｅｌ＿ｇｒｅａｔｅｒ１＿ｆｌａｇについて８つおよびｃｏｅｆｆ＿ａｂｓ＿ｌｅｖｅｌ＿ｇｒｅａｔｅｒ２＿ｆｌａｇについて１つに制限する。

【０２３４】

[0213] ４）符号情報の第４のパス。ＨＥＶＣのいくつかの例では、各非０係数の符号は、バイパスモードで第４の走査パスにおいてコーディングされる。各ＣＧについて、および基準に応じて、（逆方向走査順序における）最後非０係数（last nonzero coefficient）の符号を符号化することは、符号データヒッディング（ＳＤＨ：sign data hidding）を使用するとき、単に省略される。代わりに、符号値は、偶数が「＋」に対応し、奇数が「－」に対応するという、あらかじめ定義された規約を使用してＣＧのレベルの和のパリティ中に埋め込まれる。ＳＤＨを使用するための基準は、ＣＧの第１の非０係数と最後非０係数との間の走査順序における距離である。この距離が４に等しいかまたはそれよりも大きい場合、ＳＤＨが使用される。４のこの値は、それがＨＥＶＣテストシーケンスに対して最も大きい利得を与えるので選定された。

【０２３５】

[0214] ５）残りのビンの最後のパス。残りのビンが、さらなる走査パスにおいてコーディングされる。係数のｂａｓｅＬｅｖｅｌが次のように定義されるとする。

【０２３６】

【数２４】

$$baseLevel = \text{significant\_flag} + \text{coeff\_abs\_level\_greater1\_flag} + \text{coeff\_abs\_level\_greater2\_flag}$$

【０２３７】

[0215] ここで、フラグは、０または１の値を有し、存在しない場合、０であると推論される。その場合、係数の絶対値は単に以下の通りである。

【０２３８】

【数２５】

$$\text{absCoeffLevel} = baseLevel + \text{coeff\_abs\_level\_remaining}$$

【０２３９】

[0216] ライスパラメータ（Rice parameter）は、各ＣＧの最初に０に設定され、それは、以下のようにパラメータの前の値と現在の絶対レベルとに応じて条件付きで更新される。

【０２４０】

【数２６】

$$\text{if absCoeffLevel} > 3 \times 2m,\ m = \min(4, m+1)$$

【０２４１】

[0217] シンタックス要素ｃｏｅｆｆ＿ａｂｓ＿ｌｅｖｅｌ＿ｒｅｍａｉｎｉｎｇはバイパスモードでコーディングされ得る。さらに、ＨＥＶＣのいくつかの例は、小さい値についてはゴロムライスコード（Golomb-Rice code）を採用し、より大きい値については指数ゴロムコード（Exp-Golomb code）に切り替わる。コード間の遷移点は、一般に、単項コード長が４に等しいときである。パラメータ更新プロセスは、２値化が、分布において大きい値が観測されたとき、係数統計値に適応することを可能にする。

【０２４２】

[0218] ｉｎｔｅｒ＿ｐｒｅｄ＿ｉｄｃのコンテキストモデリング。ｉｎｔｅｒ＿ｐｒｅｄ＿ｉｄｃは、現在予測ユニットのためにｌｉｓｔ０が使用されるのか、ｌｉｓｔ１が使用されるのか、双予測が使用されるのかを指定する。シンタックス要素は、その両方がＣＡＢＡＣコンテキストコーディングされる、最高２つのビンを有する。２値化されたビンストリングは以下のように定義される。

【０２４３】
【表９】

inter__pred__idc の値	ビンストリング	ビンストリング
	(nPbW + nPbH) != 12	(nPbW + nPbH) != 12
0	00	00
1	01	01
2	1	1

【０２４４】
[0219] ここにおいて、ｎＰｂＷおよびｎＰｂＨは、それぞれ、現在ルーマ予測ブロック幅および高さを表す。
【０２４５】
[0220] 各インターコード化スライス、たとえば、ＰスライスまたはＢスライスについて、コンテキスト選択は以下のルールに基づく。
【０２４６】
－ （ｎＰｂＷ＋ｎＰｂＨ）が１２に等しくない場合、第１のビンは、４つのコンテキストを使用してコーディングされ、第２のビンは、１つのコンテキストを用いてコーディングされる。第１のビンのコンテキスト選択は現在ＣＵ深度従う。ＨＥＶＣでは、ＣＵ深度は、両端値を含む、０～３の範囲内にある。
【０２４７】
[0221] 図１１は、本開示の１つまたは複数の技法による、異なるウィンドウサイズでコンテキストベースエントロピー符号化を実行するための例示的なプロセスを示すフローチャートである。図１１の技法は、図１および図４に示されたビデオエンコーダ２０など、ビデオエンコーダによって実行され得る。説明の目的で、図１１の技法は、図１および図４のビデオエンコーダ２０のコンテキスト内で説明されるが、ビデオエンコーダ２０の構成とは異なる構成を有するビデオエンコーダが図１１の技法を実行し得る。
【０２４８】
[0222] ビデオエンコーダ２０は、コンテキストベースエントロピーコーディングを使用して符号化されるべきビンストリング（たとえば、１次元バイナリベクトル）を取得する（１１０２）。たとえば、ビデオエンコーダ２０のエントロピー符号化ユニット５６は、ビデオエンコーダ２０の予測処理ユニット４２から受信されたシンタックス要素を２値化することによってビンストリングを取得し得る。いくつかの例では、コンテキストベースエントロピーコーディングはコンテキスト適応型バイナリ算術コーディング（ＣＡＢＡＣ）を備え得る。
【０２４９】
[0223] 本開示の１つまたは複数の技法によれば、ビデオエンコーダ２０は、複数のコンテキストのうちのコンテキストのための複数のウィンドウサイズのうちのウィンドウサイズを決定する（１１０４）。いくつかの例では、ビデオエンコーダ２０は、コンテキストのための所定のウィンドウサイズに基づいてウィンドウサイズを決定し得る。いくつかの例では、ビデオエンコーダ２０は、いくつかの候補ウィンドウサイズのコーディング効率を分析することによってウィンドウサイズを決定し、最良のコーディング効率をもつ候補ウィンドウサイズをコンテキストのためのウィンドウサイズとして選択し得る。
【０２５０】
[0224] たとえば、各コンテキストについて、エントロピー符号化ユニット５６は、異なるウィンドウサイズで、記録されたビンストリングをコーディングするビットを計算し、最小ビットをもつ１つのウィンドウサイズを選択し得る。いくつかの例では、エントロピー符号化ユニット５６によって使用される異なるウィンドウサイズは、あらかじめ定義され得る。いくつかの例示的なあらかじめ定義されたウィンドウサイズは、１６、３２、６４、および１２８であるが、他のウィンドウサイズが企図される。
【０２５１】

[0225] いくつかの例では、エントロピー符号化ユニット５６は、ビットストリングを符号化するために使用されるウィンドウサイズを示す１つまたは複数のシンタックス要素を符号化し得る。たとえば、エントロピー符号化ユニット５６は、各コンテキストについて、デフォルトウィンドウサイズが使用されるのか、更新されたウィンドウサイズが使用されるのかを示す第１のフラグを、現在スライスのスライスヘッダ中で符号化し得る。一例として、エントロピー符号化ユニット５６が、デフォルトウィンドウサイズ以外のウィンドウサイズに関連するコンテキストを使用して現在スライスの１つまたは複数のビットストリングをコーディングした場合、エントロピー符号化ユニット５６は、１つまたは複数のビットストリングが、デフォルトウィンドウサイズ以外のウィンドウサイズを使用してコーディングされた現在スライスのものであることを示すために、現在スライスのスライスヘッダ中で第１のフラグを符号化し得る。同様に、エントロピー符号化ユニット５６が、デフォルトウィンドウサイズに関連するコンテキストを使用してスライスのビットストリングのすべてをコーディングした場合、エントロピー符号化ユニット５６は、現在スライスのビットストリングのすべてが、デフォルトウィンドウサイズを使用してコーディングされることを示すために、現在スライスのスライスヘッダ中で第１のフラグを符号化し得る。いくつかの例では、第１のフラグは、以下でより詳細に説明されるように、ｄｅｆａｕｌｔ＿ｕｐｄａｔｉｎｇ＿ｓｐｅｅｄ＿ｆｌａｇと呼ばれることがある。

【０２５２】

[0226] いくつかの例では、エントロピー符号化ユニット５６が、１つまたは複数のビットストリングが、デフォルトウィンドウサイズ以外のウィンドウサイズを使用してコーディングされたスライスのものであることを示す第１のフラグを、現在スライスのスライスヘッダ中で符号化する場合、エントロピー符号化ユニット５６は、現在スライスをコーディングするために使用されるコンテキストに関連するウィンドウサイズが、前にコーディングされたスライスから継承されるかどうかを示す第２のフラグを、現在スライスのスライスヘッダ中で符号化し得る。いくつかの例では、前にコーディングされたスライスは、同じスライスタイプおよび同じ初期化されたＱＰなど、現在スライスと共通する１つまたは複数のパラメータを有する最も最近コーディングされたスライスであり得る。いくつかの例では、第１のフラグは、以下でより詳細に説明されるように、ｉｎｈｅｒｉｔａｎｃｅ＿ｆｒｏｍ＿ｐｒｅｖｉｏｕｓ＿ｆｌａｇと呼ばれることがある。

シンタックス

【０２５３】

【表１０】

7.3.6 スライスセグメントヘッダシンタックス

7.3.6.1 一般的なスライスセグメントヘッダシンタックス

slice_segment_header( ) {	記述子
first_slice_segment_in_pic_flag	u(1)
if(nal_unit_type >= BLA_W_LP && nal_unit_type <= RSV_IRAP_VCL23)	
no_output_of_prior_pics_flag	u(1)
slice_pic_parameter_set_id	ue(v)
...	
if(slice_segment_header_extension_present_flag) {	
slice_segment_header_extension_length	ue(v)
for(i = 0; i < slice_segment_header_extension_length; i++)	
slice_segment_header_extension_data_byte[i]	u(8)
}	
default_updating_speed_flag	*u(1)*
if(!default_updating_speed) {	
inheritance_from_previous_flag	*u(1)*
if(!inheritance_from_previous_flag) {	
bit_map_run_length_coding ( )	
speed_index_level_coding ( )	
}	
}	
byte_alignment( )	
}	

【０２５４】

[0227] 上記で説明されたシンタックス要素のためのいくつかの例示的なセマンティクスが以下で与えられる。

【０２５５】

[0228] １に等しいｄｅｆａｕｌｔ＿ｕｐｄａｔｉｎｇ＿ｓｐｅｅｄ＿ｆｌａｇは、デフォルトウィンドウサイズがすべてのコンテキストのために使用され、ｉｎｈｅｒｉｔａｎｃｅ＿ｆｒｏｍ＿ｐｒｅｖｉｏｕｓ＿ｆｌａｇがスライスヘッダ中に存在しないことを指定し得る。０に等しいｄｅｆａｕｌｔ＿ｕｐｄａｔｉｎｇ＿ｓｐｅｅｄ＿ｆｌａｇは、ｉｎｈｅｒｉｔａｎｃｅ＿ｆｒｏｍ＿ｐｒｅｖｉｏｕｓ＿ｆｌａｇがスライスヘッダ中に存在することを指定し得る。

【０２５６】

[0229] １に等しいｉｎｈｅｒｉｔａｎｃｅ＿ｆｒｏｍ＿ｐｒｅｖｉｏｕｓ＿ｆｌａｇは、コンテキストに関連するウィンドウサイズが、同じスライスタイプと同じ初期化されたＱＰとをもつ、前にコーディングされたスライスから継承されることを指定し得る。０に等しいｉｎｈｅｒｉｔａｎｃｅ＿ｆｒｏｍ＿ｐｒｅｖｉｏｕｓ＿ｆｌａｇは、コンテキストに関連するウィンドウサイズがビットストリーム中でシグナリングされ、ｂｉｔ＿ｍａｐ＿ｒｕｎ＿ｌｅｎｇｔｈ＿ｃｏｄｉｎｇ（）およびｓｐｅｅｄ＿ｉｎｄｅｘ＿ｌｅｖｅｌ＿ｃｏｄｉｎｇが存在することを指定し得る。

【０２５７】

[0230] いくつかの例では、使用されるべきピクチャは明示的に指定され得る。ピクチャが複数のスライスを含んでいる場合、第１のスライスが使用され得るか、またはそのピ

クチャのスライスのｉｄが明示的に指定され得る。いくつかの例では、１に等しいｉｎｈｅｒｉｔａｎｃｅ＿ｆｒｏｍ＿ｐｒｅｖｉｏｕｓ＿ｆｌａｇは、コンテキストに関連するウィンドウサイズが、同じスライスタイプをもつ、前にコーディングされたスライスから継承されることを指定し得る。

【０２５８】

[0231] いくつかの例では、エントロピー符号化ユニット５６が、現在スライスをコーディングするために使用されるコンテキストに関連するウィンドウサイズが、前にコーディングされたスライスから継承されないことを示すために、第２のフラグを符号化する場合、エントロピー符号化ユニット５６は、現在スライスをコーディングするために使用されるコンテキストに関連するウィンドウサイズを示すために１つまたは複数のシンタックス要素をコーディングし得る。たとえば、エントロピー符号化ユニット５６は、異なるウィンドウサイズの使用を示す１つのビットマップ（bit map）をコーディングし、ビットが新しいウィンドウサイズを示すとき、新しいウィンドウサイズインデックス（window size index）をコーディングし得る。いくつかの例では、エントロピー符号化ユニット５６は、以下で説明されるように、現在スライスをコーディングするために使用されるコンテキストに関連するウィンドウサイズを示すために、１つまたは複数のシンタックス要素を符号化し得る。

【０２５９】

【表１１】

7.xxxx ビットマップランレンスコーディングシンタックス

bit_map_run_length_coding ( ) {	記述子
run = 0	
ctxIdx = 0	
while (ctxIdx < totalCtxNr) {	
run[i]	ue(v)
ctxIdx += run[i]	
if(ctxIdx >= totalCtxNr) {	
break;	
}	
ctxUpdatedSpeedFlag[ctxIdx] = 1	
ctxIdx ++	
}	
}	

【０２６０】

[0232] 代替的に、以下のテーブルが定義され得る。

【０２６１】

【表１２】

bit_map_run_length_coding ( ) {	記述子
run = 0	
ctxIdx = 0	
while (ctxIdx < totalCtxNr) {	
run[i]	ue(v)
ctxIdx += run[i]	
ctxUpdatedSpeedFlag[ctxIdx] = 1	
ctxIdx ++	
}	
}	

【０２６２】

【表１３】

7.xxxx 速度インデックスレベルコーディングシンタックス

speed_index_level_coding ( ) {	記述子
for (i = 0; i < totalCtxNr; i ++) {	
if (ctxUpdatedSpeedFlag[i]) {	
ctx_idx_difference[i]	ue(v)
}	
}	
}	

【０２６３】

[0233] 上記で説明されたシンタックス要素のためのいくつかの例示的なセマンティクスが以下で与えられる。

【０２６４】

[0234] ｒｕｎ［ｉ]は、デフォルト更新速度を使用する連続するコンテキストの数を示し得る。

【０２６５】

[0235] ｃｔｘＵｐｄａｔｅｄＳｐｅｅｄＦｌａｇは、ｔｏｔａｌＣｔｘＮｒ個のエントリをもつアレイであり得る。各エントリについて、それは、１つのスライスを復号する前に０であるように設定され得、これは、各コンテキストが、デフォルト確率更新速度、すなわち、デフォルトウィンドウサイズを使用することを示す。一例では、デフォルトウィンドウサイズは６４に等しい。

【０２６６】

[0236] 一例では、ｔｏｔａｌＣｔｘＮｒは、現在スライス中で使用され得るコンテキストの総数を表し得る。別の例では、ｔｏｔａｌＣｔｘＮｒは、すべてのスライス中で使用され得るコンテキストの総数を表し得る。別の例では、ｔｏｔａｌＣｔｘＮｒは、あらかじめ定義された選択されたコンテキストの総数を表し得る。

【０２６７】

[0237] ｃｔｘ＿ｉｄｘ＿ｄｉｆｆｅｒｅｎｃｅは、デフォルトウィンドウサイズと比較したウィンドウサイズのインデックスの差分を示し得る。

【０２６８】

[0238] 一例では、デフォルトウィンドウサイズは６４に等しいことがある。ｃｔｘ＿ｉｄｘ＿ｄｉｆｆｅｒｅｎｃｅが２に等しいときなど、いくつかの例では、ウィンドウサイズは１２８に設定され得る。ｃｔｘ＿ｉｄｘ＿ｄｉｆｆｅｒｅｎｃｅが０または１に等しいときなど、いくつかの例では、ウィンドウサイズは（１＜＜（ｃｔｘ＿ｉｄｘ＿ｄｉ

ｆｆｅｒｅｎｃｅ＋４））に等しく設定され得る。すなわち、４つのウィンドウサイズ、すなわち、１６、３２、６４、および１２８がサポートされ得るが、追加のまたはより少数のウィンドウサイズをもつ例が企図される。いくつかの例では、エントロピー符号化ユニット５６は、アクティブパラメータセット中で上記の情報の一部または全部をシグナリングし得る。

【０２６９】

[0239] デコーダ側において、各スライスヘッダについて、各コンテキストについてデフォルトウィンドウサイズまたは更新されたウィンドウサイズの使用を示し得る第１のフラグが最初に復号され得る。いくつかの例では、第１のフラグが１（すなわち、更新されたウィンドウサイズを使用すること）に等しい場合、前にコーディングされたピクチャからの継承、または更新されたウィンドウサイズの追加のシグナリングを示し得る、第２のフラグがさらに復号され得る。ウィンドウサイズのシグナリングが必要とされる場合、異なるウィンドウサイズの使用を示すために、ビットマップが最初にシグナリングされ、ビットが新しいウィンドウサイズを示すとき、新しいウィンドウサイズインデックスをシグナリングし得る。

【０２７０】

[0240] いずれの場合も、ビデオエンコーダ２０は、ビデオビットストリーム中でおよびコンテキストの確率状態に基づいて、ビンストリングのビンを符号化する（１１０６）。たとえば、エントロピー符号化ユニット５６は、コンテキストの最終コード化確率間隔内の確率に対する値またはポインタを表すバイナリストリームを出力し得る。

【０２７１】

[0241] ビデオエンコーダ２０は、決定されたウィンドウサイズに基づいてコンテキストモデルの確率状態を更新する（１１０８）。たとえば、ｉ番目のコンテキストモデルに関連する決定されたウィンドウサイズW_iが与えられれば、エントロピー符号化ユニット５６は、以下の式（１５）に従ってｉ番目のコンテキストモデルの確率状態を更新し得、ここで、ｋは確率の精度を表し得る。一例では、ｋは１５に等しい。

【０２７２】

【数２７】

$$P_{new}=\begin{cases}(2^k/W_i)+P_{old}-(P_{old}/W_i) & MPS\ (\text{たとえば},1)\\ P_{old}-(P_{old}/W_i) & LPS\ (\text{たとえば},(1-MPS))\end{cases} \quad (15)$$

【０２７３】

[0242] W_iが（$1<<M_i$）に等しいとき、エントロピー符号化ユニット５６によって実行される確率更新プロセスは、以下の式（１６）に示されているように書き直され得る。

【０２７４】

【数２８】

$$P_{new}=\begin{cases}P_{old}+((2^k-P_{old})>>M_i) & MPS\ (\text{たとえば},1)\\ P_{old}-(P_{old}>>M_i) & LPS\ (\text{たとえば},(1-MPS))\end{cases} \quad (16)$$

【０２７５】

[0243] ビデオエンコーダ２０は、ビデオビットストリーム中でおよびコンテキストの更新された確率状態に基づいて、別のビンを符号化する（１１０６）。いくつかの例では、符号化される他のビンは、ビンストリングの第２のビンであり得る。

【０２７６】

[0244] 図１２は、本開示の１つまたは複数の技法による、異なるウィンドウサイズでコンテキストベースエントロピー復号を実行するための例示的なプロセスを示すフローチャートである。図１２の技法は、図１および図６に示されたビデオデコーダ３０など、ビデオデコーダによって実行され得る。説明の目的で、図１２の技法は、図１および図６のビデオデコーダ３０のコンテキスト内で説明されるが、ビデオデコーダ３０の構成とは異

なる構成を有するビデオデコーダが図１２の技法を実行し得る。

【０２７７】

[0245] ビデオデコーダ３０は、ビデオビットストリームから、コンテキストベースエントロピーコーディングを使用して復号されるべきビンストリング（たとえば、１次元バイナリベクトル）を取得する（１２０２）。たとえば、ビデオデコーダ３０のエントロピー復号ユニット７０は、ビデオデータメモリ６９からビンストリングを取得し得る。いくつかの例では、コンテキストベースエントロピーコーディングはコンテキスト適応型バイナリ算術コーディング（ＣＡＢＡＣ）を備え得る。

【０２７８】

[0246] 本開示の１つまたは複数の技法によれば、ビデオデコーダ３０は、複数のコンテキストのうちのコンテキストのための複数のウィンドウサイズのうちのウィンドウサイズを決定する（１２０４）。いくつかの例では、ビデオデコーダ３０は、コンテキストのための所定のウィンドウサイズに基づいてウィンドウサイズを決定し得る。いくつかの例では、ビデオデコーダ３０は、いくつかの候補ウィンドウサイズのコーディング効率を分析することによってウィンドウサイズを決定し、最良のコーディング効率をもつ候補ウィンドウサイズをコンテキストのためのウィンドウサイズとして選択し得る。

【０２７９】

[0247] いくつかの例では、エントロピー復号ユニット７０は、ビットストリングを符号化するために使用されるウィンドウサイズを示す１つまたは複数のシンタックス要素を符号化し得る。たとえば、エントロピー復号ユニット７０は、各コンテキストについて、デフォルトウィンドウサイズが使用されるのか、更新されたウィンドウサイズが使用されるのかを示す第１のフラグを、現在スライスのスライスヘッダから復号し得る。一例として、エントロピー復号ユニット７０が、デフォルトウィンドウサイズ以外のウィンドウサイズに関連するコンテキストを使用して現在スライスの１つまたは複数のビットストリングを復号した場合、エントロピー復号ユニット７０は、１つまたは複数のビットストリングが、デフォルトウィンドウサイズ以外のウィンドウサイズを使用してコーディングされた現在スライスのものであることを示す第１のフラグを、現在スライスのスライスヘッダから復号し得る。同様に、エントロピー復号ユニット７０が、デフォルトウィンドウサイズに関連するコンテキストを使用してスライスのビットストリングのすべてを復号した場合、エントロピー復号ユニット７０は、現在スライスのビットストリングのすべてが、デフォルトウィンドウサイズを使用してコーディングされることを示す第１のフラグを、現在スライスのスライスヘッダから復号し得る。いくつかの例では、第１のフラグは、上記でより詳細に説明されたように、ｄｅｆａｕｌｔ＿ｕｐｄａｔｉｎｇ＿ｓｐｅｅｄ＿ｆｌａｇと呼ばれることがある。

【０２８０】

[0248] いくつかの例では、エントロピー符号化ユニット５６が、１つまたは複数のビットストリングが、デフォルトウィンドウサイズ以外のウィンドウサイズを使用してコーディングされたスライスのものであることを示す第１のフラグを、現在スライスのスライスヘッダ中で符号化する場合、エントロピー符号化ユニット５６は、現在スライスをコーディングするために使用されるコンテキストに関連するウィンドウサイズが、前にコーディングされたスライスから継承されるかどうかを示す第２のフラグを、現在スライスのスライスヘッダ中で符号化し得る。いくつかの例では、前にコーディングされたスライスは、同じスライスタイプおよび同じ初期化されたＱＰなど、現在スライスと共通する１つまたは複数のパラメータを有する最も最近コーディングされたスライスであり得る。いくつかの例では、第１のフラグは、図１１を参照しながら上記でより詳細に説明されたように、ｉｎｈｅｒｉｔａｎｃｅ＿ｆｒｏｍ＿ｐｒｅｖｉｏｕｓ＿ｆｌａｇと呼ばれることがある。

【０２８１】

[0249] いくつかの例では、使用されるべきピクチャは明示的に指定され得る。ピクチャが複数のスライスを含んでいる場合、第１のスライスが使用され得るか、またはそのピ

クチャのスライスのｉｄが明示的に指定され得る。いくつかの例では、１に等しいｉｎｈｅｒｉｔａｎｃｅ＿ｆｒｏｍ＿ｐｒｅｖｉｏｕｓ＿ｆｌａｇは、コンテキストに関連するウィンドウサイズが、同じスライスタイプをもつ、前にコーディングされたスライスから継承されることを指定し得る。

【０２８２】

[0250] いくつかの例では、第２のフラグが、現在スライスをコーディングするために使用されるコンテキストに関連するウィンドウサイズが、前にコーディングされたスライスから継承されないことを示す場合、エントロピー復号ユニット７０は、現在スライスをコーディングするために使用されるコンテキストに関連するウィンドウサイズを示す１つまたは複数のシンタックス要素を復号し得る。たとえば、エントロピー復号ユニット７０は、異なるウィンドウサイズの使用を示す１つのビットマップを復号し、ビットが新しいウィンドウサイズを示すとき、新しいウィンドウサイズインデックスを復号し得る。いくつかの例では、エントロピー復号ユニット７０は、図１１を参照しながら上記で説明されたように、現在スライスをコーディングするために使用されるコンテキストに関連するウィンドウサイズを示すために、１つまたは複数のシンタックス要素を復号し得る。

【０２８３】

[0251] いずれの場合も、ビデオデコーダ３０は、コンテキストの確率状態に基づいて、ビンストリングのビンを復号する（１２０６）。ビデオデコーダ３０は、決定されたウィンドウサイズと復号されたビンとに基づいてコンテキストモデルの確率状態を更新する（１２０８）。たとえば、ｉ番目のコンテキストモデルに関連する決定されたウィンドウサイズW_iが与えられれば、エントロピー復号ユニット７０は、上記の式（１５）に従ってｉ番目のコンテキストモデルの確率状態を更新し得る。

【０２８４】

[0252] ビデオデコーダ３０は、コンテキストの更新された確率状態に基づいて、別のビンを復号する（１２０６）。いくつかの例では、符号化される他のビンは、ビンストリングの第２のビンであり得る。

【０２８５】

[0253] 以下の番号付けされた例は、本開示の１つまたは複数の態様を示し得る。

【０２８６】

[0254] 例１．ビデオデータのエントロピーコーディングのための方法であって、本方法が、ビデオデータのシンタックス要素のための値をエントロピーコーディングするために、コンテキスト適応型エントロピーコーディングプロセスにおいて使用される複数のコンテキストのうちのコンテキストのための複数のウィンドウサイズのうちのウィンドウサイズを決定することと、コンテキストの確率状態に基づいて、シンタックス要素のための値のビンをエントロピーコーディングすることと、ウィンドウサイズとコーディングされたビンとに基づいてコンテキストの確率状態を更新することとを備える、方法。

【０２８７】

[0255] 例２．コンテキスト適応型エントロピーコーディングプロセスが、コンテキスト適応型バイナリ算術コーディング（ＣＡＢＡＣ）プロセス、またはコンテキスト適応型可変長コーディング（ＣＡＶＬＣ）プロセスを備える、請求項１に記載の方法。

【０２８８】

[0256] 例３．更新された確率状態に基づいて、同じコンテキストに関連する別のビンをエントロピーコーディングすることをさらに備える、請求項１に記載の方法。

【０２８９】

[0257] 例４．コンテキストが第１のコンテキストであり、本方法が、複数のコンテキストのうちの第２のコンテキストのための複数のウィンドウサイズのうちのウィンドウサイズを決定することをさらに備え、ここにおいて、第２のコンテキストのウィンドウサイズが第１のコンテキストのウィンドウサイズとは異なる、請求項１に記載の方法。

【０２９０】

[0258] 例５．第１のコンテキストのためのウィンドウサイズと第２のコンテキストの

ためのウィンドウサイズとが、コーディングされたビンを含むビットストリーム中でシグナリングされない（not signaled）、請求項４に記載の方法。

【０２９１】

[0259] 例６．複数のウィンドウサイズが、ウィンドウサイズのあらかじめ定義されたセットを備える、請求項１に記載の方法。

【０２９２】

[0260] 例７．エントロピーコーディングすることがエントロピー符号化することを備え、ここにおいて、ウィンドウサイズを決定することが、ウィンドウサイズのあらかじめ定義されたセットのそれぞれのウィンドウサイズについて、シンタックス要素のためのビン値を含む特定のビンストリングをエントロピー符号化するために使用されるビットのそれぞれの量を決定することと、特定のビンストリングをエントロピー符号化するために、ビットの最も小さい量に対応する、ウィンドウサイズのあらかじめ定義されたセットのうちのウィンドウサイズをコンテキストのためのウィンドウサイズとして選択することとを備える、請求項６に記載の方法。

【０２９３】

[0261] 例８．デフォルトウィンドウサイズが複数のコンテキストのために使用されるかどうかを示す第１のシンタックス要素をコーディングすることをさらに備える、請求項１に記載の方法。

【０２９４】

[0262] 例９．デフォルトウィンドウサイズが複数のコンテキストのために使用されないことを示す第１のシンタックス要素に基づいて、コンテキストのためのウィンドウサイズを示す第２のシンタックス要素をコーディングすることをさらに備える、請求項８に記載の方法。

【０２９５】

[0263] 例１０．コンテキストのためのウィンドウサイズを示すために、第２のシンタックス要素が、コンテキストのためのウィンドウサイズとデフォルトウィンドウサイズとの間の差分を示す、請求項９に記載の方法。

【０２９６】

[0264] 例１１．第１のシンタックス要素をコーディングすることが、第１のシンタックス要素を含む現在スライスのスライスヘッダをコーディングすることを備え、ここにおいて、第１のシンタックス要素は、現在スライスのビンをエントロピーコーディングするとき、デフォルトウィンドウサイズが複数のコンテキストのために使用されるかどうかを示す、請求項８に記載の方法。

【０２９７】

[0265] 例１２．現在スライスのスライスヘッダ中で、複数のコンテキストのためのウィンドウサイズが、前にコーディングされたスライスから継承されるかどうかを示すシンタックス要素をコーディングすることをさらに備える、請求項１に記載の方法。

【０２９８】

[0266] 例１３．コンテキストのためのウィンドウサイズを決定することが、シンタックス要素のタイプに基づいて、コンテキストのためのウィンドウサイズを決定することを備える、請求項１に記載の方法。

【０２９９】

[0267] 例１４．エントロピーコーディングすることがエントロピー復号することを備え、本方法が、コード化ビデオビットストリーム（coded video bitstream）から、コンテキストのためのウィンドウサイズを示す１つまたは複数のシンタックス要素を復号することをさらに備える、請求項１に記載の方法。

【０３００】

[0268] 例１５．ビデオデータのエントロピーコーディングのための装置であって、本装置が、ビデオデータのシンタックス要素のための値をエントロピーコーディングするために、コンテキスト適応型エントロピーコーディングプロセスにおいて使用される複数の

コンテキストを記憶するように構成されたメモリと、例１から１４の任意の組合せに記載の方法を実行するように構成された１つまたは複数のプロセッサとを備える、装置。

【０３０１】

[0269] 例１６．本装置が、集積回路、マイクロプロセッサ、またはワイヤレス通信デバイス（wireless communication device）のうちの少なくとも１つを備える、例１５に記載の装置。

【０３０２】

[0270] 例１７．復号ビデオデータを表示するように構成されたディスプレイ（display）をさらに備える、例１５から１６の任意の組合せに記載の装置。

【０３０３】

[0271] 例１８．ビデオデータをキャプチャするように構成されたカメラ（camera）をさらに備える、例１５から１７の任意の組合せに記載の装置。

【０３０４】

[0272] 例１９．ビデオデータのエントロピーコーディングのための装置であって、本装置が、例１から１４の任意の組合せに記載の方法を実行するための手段を備える、装置。

【０３０５】

[0273] 例２０．実行されたとき、ビデオコーディングデバイスの１つまたは複数のプロセッサに、例１から１４の任意の組合せに記載の方法を実行させる命令を記憶するコンピュータ可読記憶媒体。

【０３０６】

[0274] 例２１．ビデオ復号デバイスによって処理されたとき、ビデオ復号デバイスの１つまたは複数のプロセッサに、シンタックス要素のための値をエントロピーコーディングするために、コンテキスト適応型コーディングプロセスにおいて使用される複数のコンテキストのうちのコンテキストのための複数のウィンドウサイズのうちのウィンドウサイズを決定することと、コンテキストの確率状態に基づいて、シンタックス要素のための値のビンをエントロピーコーディングすることと、ウィンドウサイズとコーディングされたビンとに基づいてコンテキストの確率状態を更新することと、コンテキストモデルの更新された確率状態に基づいて、同じコンテキストをもつ次のビンをエントロピーコーディングすることとを行わせるビデオデータを記憶するコンピュータ可読記憶媒体。

【０３０７】

[0275] 例２２．１つまたは複数のプロセッサに、例１から１４の任意の組合せに記載の方法を実行させる命令をさらに記憶する、例２１に記載のコンピュータ可読記憶媒体。

【０３０８】

[0276] １つまたは複数の例では、説明された機能は、ハードウェア、ソフトウェア、ファームウェア、またはそれらの任意の組合せで実装され得る。ソフトウェアで実装される場合、機能は、１つまたは複数の命令またはコードとしてコンピュータ可読媒体上に記憶されるか、あるいはコンピュータ可読媒体を介して送信され、ハードウェアベースの処理ユニットによって実行され得る。コンピュータ可読媒体は、データ記憶媒体などの有形媒体に対応する、コンピュータ可読記憶媒体を含み得るか、または、たとえば、通信プロトコルに従って、ある場所から別の場所へのコンピュータプログラムの転送を可能にする任意の媒体を含む通信媒体を含み得る。このようにして、コンピュータ可読媒体は、概して、（１）非一時的である有形コンピュータ可読記憶媒体、あるいは（２）信号または搬送波などの通信媒体に対応し得る。データ記憶媒体は、本開示で説明された技法の実装のための命令、コードおよび／またはデータ構造を取り出すために、１つまたは複数のコンピュータあるいは１つまたは複数のプロセッサによってアクセスされ得る、任意の利用可能な媒体であり得る。コンピュータプログラム製品はコンピュータ可読媒体を含み得る。

【０３０９】

[0277] 限定ではなく例として、そのようなコンピュータ可読記憶媒体は、ＲＡＭ、ＲＯＭ、ＥＥＰＲＯＭ（登録商標）、ＣＤ－ＲＯＭまたは他の光ディスクストレージ、磁気

ディスクストレージ、または他の磁気ストレージデバイス、フラッシュメモリ、あるいは命令またはデータ構造の形態の所望のプログラムコードを記憶するために使用され得、コンピュータによってアクセスされ得る、任意の他の媒体を備えることができる。また、いかなる接続もコンピュータ可読媒体と適切に呼ばれる。たとえば、命令が、同軸ケーブル、光ファイバーケーブル、ツイストペア、デジタル加入者回線（ＤＳＬ）、または赤外線、無線、およびマイクロ波などのワイヤレス技術を使用して、ウェブサイト、サーバ、または他のリモートソースから送信される場合、同軸ケーブル、光ファイバーケーブル、ツイストペア、ＤＳＬ、または赤外線、無線、およびマイクロ波などのワイヤレス技術は、媒体の定義に含まれる。ただし、コンピュータ可読記憶媒体およびデータ記憶媒体は、接続、搬送波、信号、または他の一時的媒体を含まないが、代わりに非一時的有形記憶媒体を対象とすることを理解されたい。本明細書で使用されるディスク（disk）およびディスク（disc）は、コンパクトディスク（disc）（ＣＤ）、レーザーディスク（登録商標）（disc）、光ディスク（disc）、デジタル多用途ディスク（disc）（ＤＶＤ）、フロッピー（登録商標）ディスク（disk）およびＢｌｕ－ｒａｙディスク（disc）を含み、ここで、ディスク（disk）は、通常、データを磁気的に再生し、ディスク（disc）は、データをレーザーで光学的に再生する。上記の組合せもコンピュータ可読媒体の範囲内に含まれるべきである。

【０３１０】

[0278] 命令は、１つまたは複数のデジタル信号プロセッサ（ＤＳＰ）、汎用マイクロプロセッサ、特定用途向け集積回路（ＡＳＩＣ）、フィールドプログラマブル論理アレイ（ＦＰＧＡ）、あるいは他の等価な集積回路またはディスクリート論理回路など、１つまたは複数のプロセッサによって実行され得る。したがって、本明細書で使用される「プロセッサ」という用語は、上記の構造、または本明細書で説明された技法の実装に好適な他の構造のいずれかを指すことがある。さらに、いくつかの態様では、本明細書で説明された機能は、符号化および復号のために構成された専用ハードウェアおよび／またはソフトウェアモジュール内に与えられるか、あるいは複合コーデックに組み込まれ得る。また、本技法は、１つまたは複数の回路または論理要素で十分に実装され得る。

【０３１１】

[0279] 本開示の技法は、ワイヤレスハンドセット、集積回路（ＩＣ：integrated circuit）またはＩＣのセット（たとえば、チップセット）を含む、多種多様なデバイスまたは装置で実装され得る。本開示では、開示される技法を実行するように構成されたデバイスの機能的態様を強調するために、様々な構成要素、モジュール、またはユニットが説明されたが、それらの構成要素、モジュール、またはユニットは、必ずしも異なるハードウェアユニットによる実現を必要とするとは限らない。むしろ、上記で説明されたように、様々なユニットが、好適なソフトウェアおよび／またはファームウェアとともに、上記で説明された１つまたは複数のプロセッサを含めて、コーデックハードウェアユニットにおいて組み合わせられるか、または相互動作可能なハードウェアユニットの集合によって与えられ得る。

【０３１２】

[0280] 様々な例が説明された。これらおよび他の例は以下の特許請求の範囲内に入る。

以下に本願の出願当初の特許請求の範囲に記載された発明を付記する。

[Ｃ１] ビデオデータのエントロピーコーディングのための方法であって、前記方法が、

前記ビデオデータのシンタックス要素のための値をエントロピーコーディングするために、コンテキスト適応型エントロピーコーディングプロセスにおいて使用される複数のコンテキストのうちのコンテキストのための複数のウィンドウサイズのうちのウィンドウサイズを決定することと、

前記コンテキストの確率状態に基づいて、前記シンタックス要素のための前記値のビンをエントロピーコーディングすることと、

前記ウィンドウサイズと前記コーディングされたビンとに基づいて前記コンテキストの

前記確率状態を更新することとを備える、方法。
[Ｃ２] 前記コンテキスト適応型エントロピーコーディングプロセスが、コンテキスト適応型バイナリ算術コーディング（ＣＡＢＡＣ）プロセス、またはコンテキスト適応型可変長コーディング（ＣＡＶＬＣ）プロセスを備える、Ｃ１に記載の方法。
[Ｃ３] 前記更新された確率状態に基づいて、同じコンテキストに関連する別のビンをエントロピーコーディングすることをさらに備える、Ｃ１に記載の方法。
[Ｃ４] 前記コンテキストが第１のコンテキストであり、前記方法が、
前記複数のコンテキストのうちの第２のコンテキストのための前記複数のウィンドウサイズのうちのウィンドウサイズを決定することをさらに備え、ここにおいて、前記第２のコンテキストの前記ウィンドウサイズが前記第１のコンテキストの前記ウィンドウサイズとは異なる、Ｃ１に記載の方法。
[Ｃ５] 前記第１のコンテキストのための前記ウィンドウサイズと前記第２のコンテキストのための前記ウィンドウサイズとが、前記コーディングされたビンを含むビットストリーム中でシグナリングされない、Ｃ４に記載の方法。
[Ｃ６] 前記複数のウィンドウサイズが、ウィンドウサイズのあらかじめ定義されたセットを備える、Ｃ１に記載の方法。
[Ｃ７] エントロピーコーディングすることがエントロピー符号化することを備え、ここにおいて、前記ウィンドウサイズを決定することが、
ウィンドウサイズの前記あらかじめ定義されたセットのそれぞれのウィンドウサイズについて、前記シンタックス要素のための前記ビン値を含む特定のビンストリングをエントロピー符号化するために使用されるビットのそれぞれの量を決定することと、
前記特定のビンストリングをエントロピー符号化するために、ビットの最も小さい量に対応する、ウィンドウサイズの前記あらかじめ定義されたセットのうちの前記ウィンドウサイズを前記コンテキストのための前記ウィンドウサイズとして選択することとを備える、Ｃ６に記載の方法。
[Ｃ８] デフォルトウィンドウサイズが前記複数のコンテキストのために使用されるかどうかを示す第１のシンタックス要素をコーディングすることをさらに備える、Ｃ１に記載の方法。
[Ｃ９] 前記デフォルトウィンドウサイズが前記複数のコンテキストのために使用されないことを示す前記第１のシンタックス要素に基づいて、前記コンテキストのための前記ウィンドウサイズを示す第２のシンタックス要素をコーディングすることをさらに備える、Ｃ８に記載の方法。
[Ｃ１０] 前記コンテキストのための前記ウィンドウサイズを示すために、前記第２のシンタックス要素が、前記コンテキストのための前記ウィンドウサイズと前記デフォルトウィンドウサイズとの間の差分を示す、Ｃ９に記載の方法。
[Ｃ１１] 前記第１のシンタックス要素をコーディングすることが、前記第１のシンタックス要素を含む現在スライスのスライスヘッダをコーディングすることを備え、ここにおいて、前記第１のシンタックス要素は、前記現在スライスのビンをエントロピーコーディングするとき、前記デフォルトウィンドウサイズが前記複数のコンテキストのために使用されるかどうかを示す、Ｃ８に記載の方法。
[Ｃ１２] 現在スライスのスライスヘッダ中で、前記複数のコンテキストのためのウィンドウサイズが、前にコーディングされたスライスから継承されるかどうかを示すシンタックス要素をコーディングすることをさらに備える、Ｃ１に記載の方法。
[Ｃ１３] 前記コンテキストのための前記ウィンドウサイズを決定することが、
前記シンタックス要素のタイプに基づいて、前記コンテキストのための前記ウィンドウサイズを決定することを備える、Ｃ１に記載の方法。
[Ｃ１４] エントロピーコーディングすることがエントロピー復号することを備え、前記方法が、
コード化ビデオビットストリームから、前記コンテキストのための前記ウィンドウサイズを示す１つまたは複数のシンタックス要素を復号することをさらに備える、Ｃ１に記載

の方法。
[Ｃ１５] ビデオデータのエントロピーコーディングのための装置であって、前記装置が、
前記ビデオデータのシンタックス要素のための値をエントロピーコーディングするために、コンテキスト適応型エントロピーコーディングプロセスにおいて使用される複数のコンテキストを記憶するように構成されたメモリと、
１つまたは複数のプロセッサとを備え、前記１つまたは複数のプロセッサが、
前記複数のコンテキストのうちのコンテキストのための複数のウィンドウサイズのうちのウィンドウサイズを決定することと、
コンテキストモデルの確率状態に基づいて、前記シンタックス要素のための前記値のビンをエントロピーコーディングすることと、
前記ウィンドウサイズ前記コーディングされたビンに基づいて前記コンテキストモデルの前記確率状態を更新することと
を行うように構成された、装置。
[Ｃ１６] 前記１つまたは複数のプロセッサが、
前記更新された確率状態に基づいて、同じコンテキストに関連する別のビンをエントロピーコーディングするようにさらに構成された、Ｃ１５に記載の装置。
[Ｃ１７] 前記コンテキストモデルが第１のコンテキストであり、ここにおいて、前記１つまたは複数のプロセッサが、
前記複数のコンテキストのうちの第２のコンテキストのための前記複数のウィンドウサイズのうちのウィンドウサイズを決定するようにさらに構成され、ここにおいて、前記第２のコンテキストの前記ウィンドウサイズが前記第１のコンテキストの前記ウィンドウサイズとは異なる、Ｃ１６に記載の装置。
[Ｃ１８] 前記第１のコンテキストのための前記ウィンドウサイズと前記第２のコンテキストのための前記ウィンドウサイズとが、前記コーディングされたビンを含むビットストリーム中でシグナリングされない、Ｃ１７に記載の装置。
[Ｃ１９] 前記複数のウィンドウサイズが、ウィンドウサイズのあらかじめ定義されたセットを備える、Ｃ１６に記載の装置。
[Ｃ２０] エントロピーコーディングするために、前記１つまたは複数のプロセッサがエントロピー符号化するように構成され、ここにおいて、前記ウィンドウサイズを決定するために、前記１つまたは複数のプロセッサが、
ウィンドウサイズの前記あらかじめ定義されたセットのそれぞれのウィンドウサイズについて、前記シンタックス要素のための前記ビン値を含む特定のビンストリングをエントロピー符号化するために使用されるビットのそれぞれの量を決定することと、
前記特定のビンストリングをエントロピー符号化するために、前記コンテキストのための前記ウィンドウサイズとして、ビットの最も小さい量に対応する、ウィンドウサイズの前記あらかじめ定義されたセットのうちの前記ウィンドウサイズを選択することとを行うように構成された、Ｃ１９に記載の装置。
[Ｃ２１] 前記１つまたは複数のプロセッサが、
デフォルトウィンドウサイズが前記複数のコンテキストのために使用されるかどうかを示す第１のシンタックス要素をコーディングするようにさらに構成された、Ｃ１５に記載の装置。
[Ｃ２２] 前記デフォルトウィンドウサイズが前記複数のコンテキストのために使用されないことを示す前記第１のシンタックス要素に基づいて、前記１つまたは複数のプロセッサが、
前記コンテキストのための前記ウィンドウサイズを示す第２のシンタックス要素をコーディングするようにさらに構成された、Ｃ２１に記載の装置。
[Ｃ２３] 前記コンテキストのための前記ウィンドウサイズを示すために、前記第２のシンタックス要素が、前記コンテキストのための前記ウィンドウサイズと前記デフォルトウィンドウサイズとの間の差分を示す、Ｃ２２に記載の装置。

[Ｃ２４] 前記第１のシンタックス要素をコーディングするために、前記１つまたは複数のプロセッサが、前記第１のシンタックス要素を含む現在スライスのスライスヘッダをコーディングするように構成され、ここにおいて、前記第１のシンタックス要素は、前記現在スライスのビンをエントロピーコーディングするとき、前記デフォルトウィンドウサイズが複数のコンテキストモデルのために使用されるかどうかを示す、Ｃ２１に記載の装置。

[Ｃ２５] 前記１つまたは複数のプロセッサが、

現在スライスのスライスヘッダ中で、前記複数のコンテキストのためのウィンドウサイズが、前にコーディングされたスライスから継承されるかどうかを示すシンタックス要素をコーディングするようにさらに構成された、Ｃ１５に記載の装置。

[Ｃ２６] 前記コンテキストのための前記ウィンドウサイズを決定するために、前記１つまたは複数のプロセッサが、

前記シンタックス要素のタイプに基づいて、前記コンテキストのための前記ウィンドウサイズを決定するように構成された、Ｃ１５に記載の装置。

[Ｃ２７] 前記装置が、

集積回路、

マイクロプロセッサ、または

ワイヤレス通信デバイスのうちの少なくとも１つを備える、Ｃ１５に記載の装置。

[Ｃ２８] 復号ビデオデータを表示するように構成されたディスプレイをさらに備える、Ｃ２７に記載の装置。

[Ｃ２９] 前記ビデオデータをキャプチャするように構成されたカメラをさらに備える、２７　１５に記載の装置。

[Ｃ３０] エントロピーコーディングするために、前記１つまたは複数のプロセッサが、前記シンタックス要素の前記値をエントロピー復号するように構成された、Ｃ１５に記載の装置。

[Ｃ３１] ビデオデータのエントロピーコーディングのための装置であって、前記装置が、

前記ビデオデータのシンタックス要素のための値をエントロピーコーディングするために、コンテキスト適応型コーディングプロセスにおいて使用される複数のコンテキストのうちのコンテキストのための複数のウィンドウサイズのうちのウィンドウサイズを決定するための手段と、

前記コンテキストの確率状態に基づいて、前記シンタックス要素のための前記値のビンをエントロピーコーディングするための手段と、

前記ウィンドウサイズと前記コーディングされたビンとに基づいて前記コンテキストの前記確率状態を更新するための手段とを備える、装置。

[Ｃ３２] 実行されたとき、ビデオコーディングデバイスの１つまたは複数のプロセッサに、

ビデオデータのシンタックス要素のための値をエントロピーコーディングするために、コンテキスト適応型コーディングプロセスにおいて使用される複数のコンテキストのうちのコンテキストのための複数のウィンドウサイズのうちのウィンドウサイズを決定することと、

前記コンテキストの確率状態に基づいて、前記シンタックス要素のための前記値のビンをエントロピーコーディングすることと、

前記ウィンドウサイズと前記コーディングされたビンとに基づいて前記コンテキストの前記確率状態を更新することとを行わせる命令を記憶するコンピュータ可読記憶媒体。

【図１】

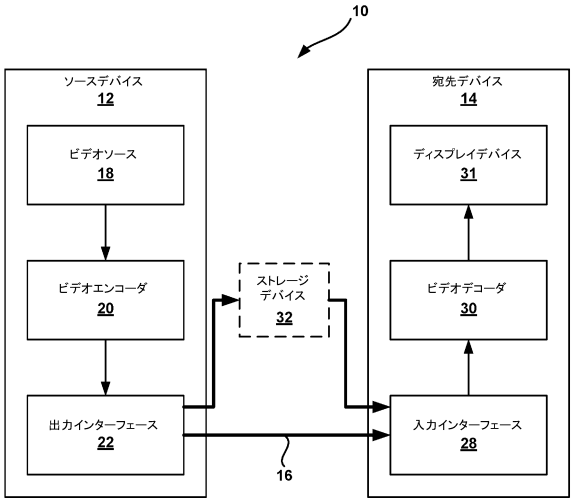

FIG. 1

【図２Ａ】

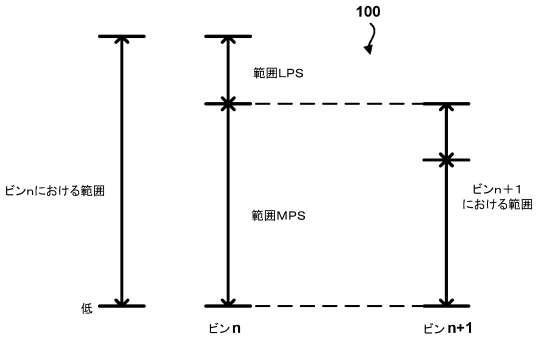

FIG. 2A

【図２Ｂ】

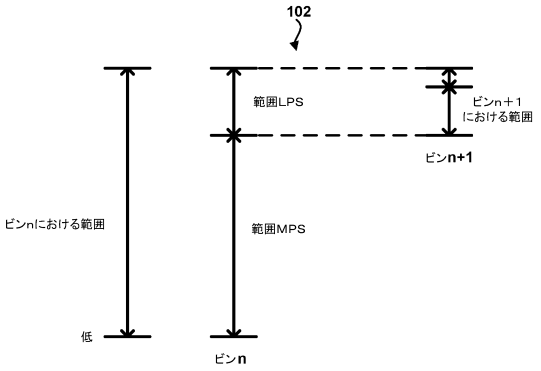

FIG. 2B

【図３】

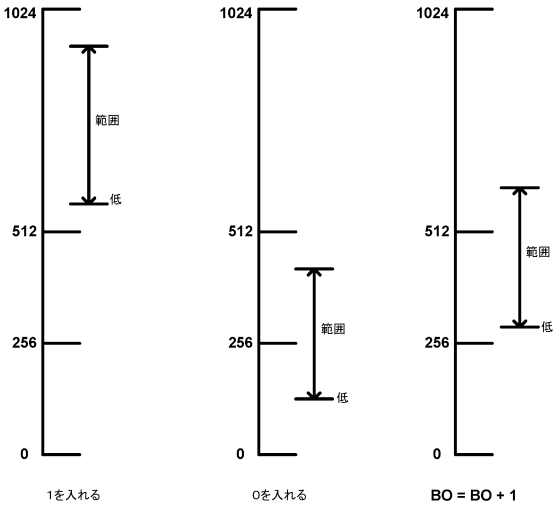

FIG. 3

【図４】

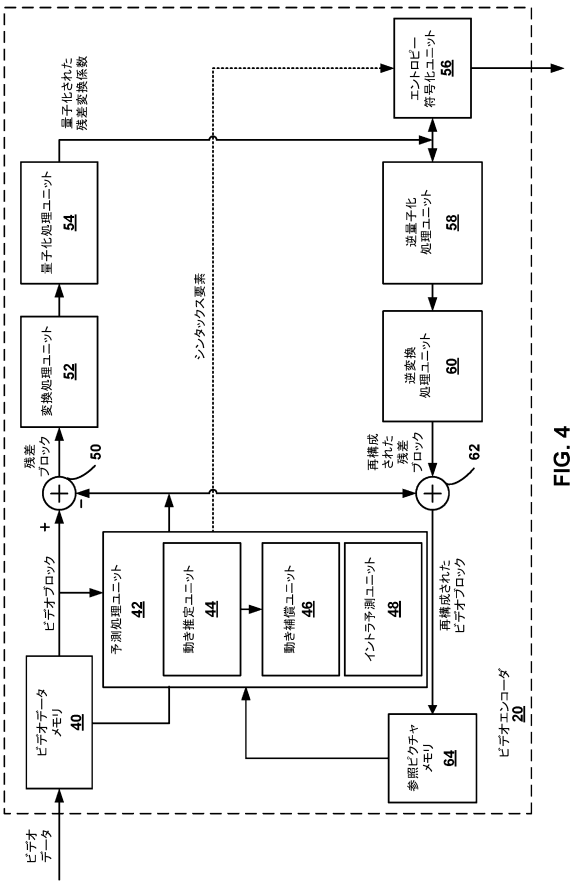

FIG. 4

【図５】

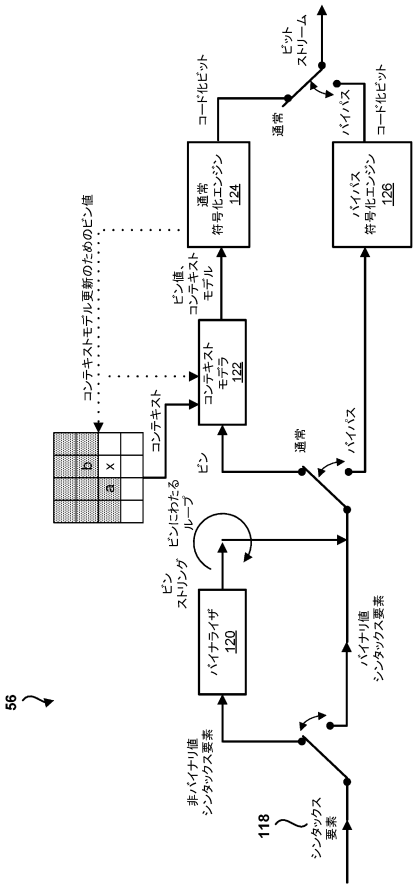

FIG. 5

【図６】

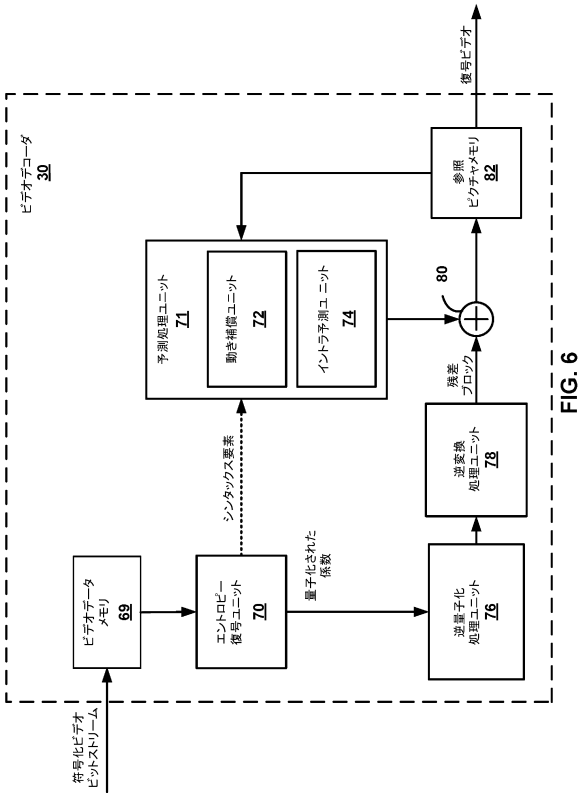

FIG. 6

【図７】

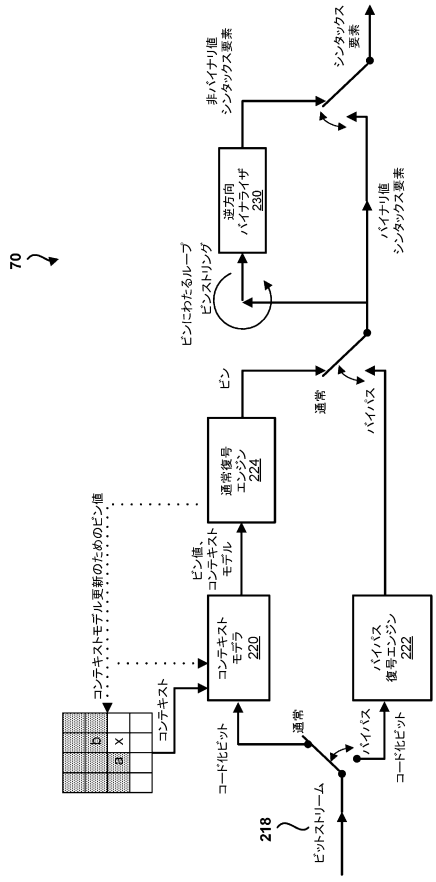

FIG. 7

【図８】

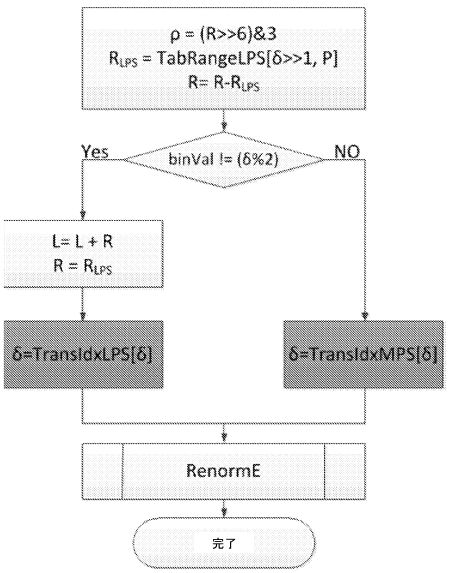

FIG. 8

【図９】

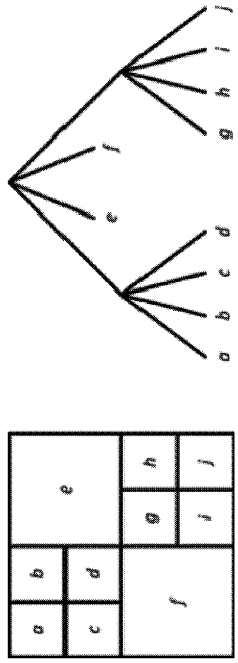

FIG. 9

【図１０】

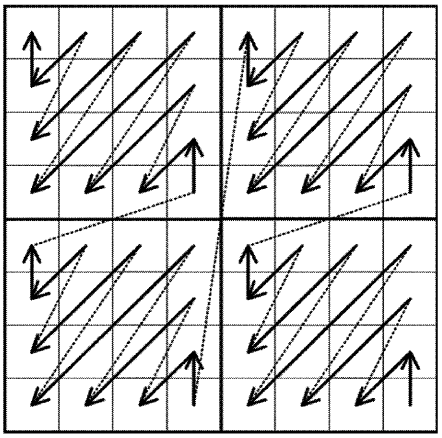

FIG. 10

【図１１】

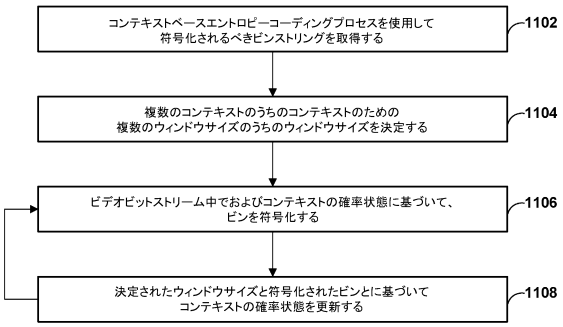

FIG. 11

【図１２】

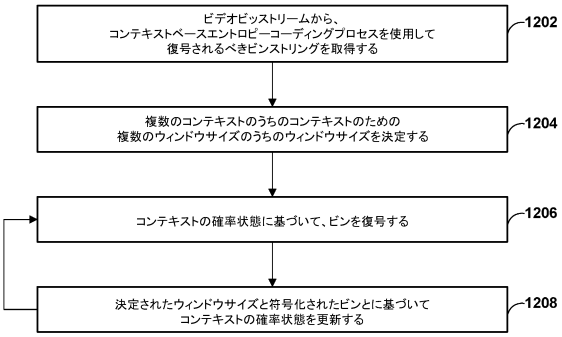

FIG. 12

フロントページの続き

(72)発明者　ジャン、リ
アメリカ合衆国、カリフォルニア州　９２１２１－１７１４、サン・ディエゴ、モアハウス・ドライブ　５７７５
(72)発明者　チェン、ジャンレ
アメリカ合衆国、カリフォルニア州　９２１２１－１７１４、サン・ディエゴ、モアハウス・ドライブ　５７７５
(72)発明者　ジャオ、シン
アメリカ合衆国、カリフォルニア州　９２１２１－１７１４、サン・ディエゴ、モアハウス・ドライブ　５７７５
(72)発明者　リ、シャン
アメリカ合衆国、カリフォルニア州　９２１２１－１７１４、サン・ディエゴ、モアハウス・ドライブ　５７７５
(72)発明者　リウ、ホンビン
中華人民共和国、１０２２０８　ベイジン、チャンピン・ディストリクト、フイロングアン、ロンフアユアン・セカンド・セクション、サーティーセブンス・ビルディング、ルーム　９－２０２
(72)発明者　チェン、イン
アメリカ合衆国、カリフォルニア州　９２１２１－１７１４、サン・ディエゴ、モアハウス・ドライブ　５７７５
(72)発明者　カルチェビチ、マルタ
アメリカ合衆国、カリフォルニア州　９２１２１－１７１４、サン・ディエゴ、モアハウス・ドライブ　５７７５

審査官　富樫　明

(56)参考文献　特表２０１４－５２３１８６（ＪＰ，Ａ）
特表２０１４－５１５８９４（ＪＰ，Ａ）
特開２００４－１２９２０６（ＪＰ，Ａ）
国際公開第２０１３／０５７７８３（WO，Ａ１）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｈ０４Ｎ　１９／００－１９／９８