

(19) 日本国特許庁(JP)

(12) 特 許 公 報(B2)

(11) 特許番号

特許第6560170号

(P6560170)

(45) 発行日 令和1年8月14日(2019.8.14)

(24) 登録日 令和1年7月26日(2019.7.26)

(51) Int.Cl. F I
HO 4 N 19/70 (2014.01) HO 4 N 19/70
HO 4 N 19/13 (2014.01) HO 4 N 19/13
HO 4 N 19/436 (2014.01) HO 4 N 19/436

請求項の数 7 (全 53 頁)

(21) 出願番号	特願2016-160844 (P2016-160844)	(73) 特許権者	515089080
(22) 出願日	平成28年8月18日 (2016.8.18)		ジーイー ビデオ コンプレッション エルエルシー
(62) 分割の表示	特願2015-504979 (P2015-504979) の分割		アメリカ合衆国 ニューヨーク州 122 11 オールバニー サウスウッズ プールバード 8
原出願日	平成25年4月15日 (2013.4.15)	(74) 代理人	100079577
(65) 公開番号	特開2017-22724 (P2017-22724A)		弁理士 岡田 全啓
(43) 公開日	平成29年1月26日 (2017.1.26)	(72) 発明者	シーアル トーマス
審査請求日	平成28年8月22日 (2016.8.22)		ドイツ連邦共和国 10437 ベルリン
(31) 優先権主張番号	61/624,098		ドゥンカーシュトラーク 72
(32) 優先日	平成24年4月13日 (2012.4.13)	(72) 発明者	ジョージ ヴァレーリ
(33) 優先権主張国・地域又は機関	米国 (US)		ドイツ連邦共和国 10365 ベルリン
(31) 優先権主張番号	61/666,185		ヨーン-ジーク-シュトラーク 24
(32) 優先日	平成24年6月29日 (2012.6.29)		
(33) 優先権主張国・地域又は機関	米国 (US)		最終頁に続く

(54) 【発明の名称】 低遅延画像符号化

(57) 【特許請求の範囲】

【請求項 1】

C T B (符号化ツリーブロック) のアレイに再分割されている画像 (10) を、前記画像 (10) がパーティション分割されているスライス (14) 単位で、前記画像が符号化されているデータストリーム (12) から、W P P 処理を用いて、再構成するためのデコーダであって、前記デコーダは、スライス順序 (16) に従って前記データストリーム (12) から前記スライス (14) を復号化するように構成され、それぞれのスライス (14) は、前記画像 (10) 内における前記それぞれのスライスの復号化開始点が存在する C T B を、C T B ラスタスキャン順に沿って、アドレス指定するスライスアドレス (400) を含み、さらに、前記デコーダは、

前記スライスのスライスアドレスを用いて、C T B ラスタスキャン順における C T B 列の第 1 のシンタックス要素を含むスライスを識別することによって、前記スライスがグループ化された W P P サブストリームのエントリ・ポイントを識別して、

前記 C T B ラスタスキャン順序を用いて復号化を実行するように構成され、所定の C T B の解析を開始する際に C T B 単位で測定した、所定の C T B の C T B アドレスモジュロ画像の幅の整数倍が、C T B 単位で測定した、前記所定の C T B の C T B アドレスモジュロ画像の幅の整数倍が 0 であるか 2 であるかをチェックし、C T B 単位で測定した、前記所定の C T B の C T B アドレスモジュロ画像の幅の整数倍が 2 であれば、コンテキスト表における C A B A C コンテキスト変数の現在の状態を記憶し、C T B 単位で測定した、前記所定の C T B の C T B アドレスモジュロ画像の幅の整数倍が 0 であれば、前記コンテキ

スト表を使って前記C A B A Cコンテキスト変数を初期化し、

前記スライス順序に従って、千鳥状に前記W P Pサブストリームの前記復号化を連続的に開始するとともに前記W P Pサブストリームを並列復号化するように構成される、デコーダ。

【請求項2】

前記データストリームは、イントラ予測またはインター予測ならびに変換符号化のために前記画像の前記C T Bをブロックに分割したことを明らかにする分割情報を有し、前記分割情報はルマおよびクロマについて異なる分割がなされていることを明らかにする、請求項1に記載のデコーダ。

【請求項3】

C T B（符号化ツリーブロック）のアレイに再分割された画像（10）を、前記画像（10）がパーティション分割されているスライス（14）単位で、W P P処理を用いて、データストリーム（12）に符号化するためのエンコーダであって、前記エンコーダは、スライス順序（16）に従って前記スライス（14）を前記データストリーム（12）に符号化するように構成され、前記エンコーダは、それぞれのスライス（14）に対し、前記画像（10）内における前記それぞれのスライスの符号化開始点があるC T Bを、C T Bラスタスキャン順に沿って、アドレス指定するスライスアドレス（400）を提供するように構成され、さらに、前記エンコーダは、

それぞれのW P Pサブストリームについて、各W P Pサブストリームのスライス順序において最初のスライスが、C T Bラスタスキャン順においてC T B列の第1のシンタックス要素を含むように、前記スライスをW P Pサブストリームにグループ化し、さらに

前記データストリームが前記C T Bラスタスキャン順を用いて復号可能であるように、前記スライス順序に従って、且つ、所定のC T Bの解析を開始する際に、C T B単位で測定した前記所定のC T Bの前記C T Bアドレスモジュロ前記画像の幅の整数倍数が0であるか2であるかをチェックし、C T B単位で測定した前記所定のC T Bの前記C T Bアドレスモジュロ前記画像の前記幅の前記整数倍が2であれば、コンテキスト表におけるC A B A Cコンテキスト変数の現在の状態を記憶し、C T B単位で測定した、前記所定のC T Bの前記C T Bアドレスモジュロ前記画像の前記幅の前記整数倍が0であれば、前記コンテキスト表を用いて前記C A B A Cコンテキスト変数を初期化することによって、千鳥状に前記W P Pサブストリームの前記符号化を連続的に開始するとともに、前記W P Pサブストリームを並列符号化するように構成される、エンコーダ。

【請求項4】

前記データストリームは、イントラ予測またはインター予測ならびに変換符号化のために前記画像のC T Bがブロックに分割されたことを明らかにする分割情報を有し、前記分割情報はルマおよびクロマについて異なる分割が行われていることを明らかにする、請求項3に記載のエンコーダ。

【請求項5】

C T B（符号化ツリーブロック）のアレイに再分割された画像（10）を、前記画像（10）がパーティション分割されているスライス（14）単位で前記画像が符号化されているデータストリーム（12）から、W P P処理を用いて、再構成するための方法であって、前記方法は、スライス順序（16）に従って前記データストリーム（12）から前記スライス（14）を復号化するステップであって、それぞれのスライス（14）は、前記画像（10）内における前記それぞれのスライスの復号化開始点があるC T BをC T Bラスタスキャン順でアドレス指定するスライスアドレス（400）を含むステップを含み、さらに、前記方法は、

前記スライスがグループ化されたW P Pサブストリームのエントリ・ポイントを、前記スライスのスライスアドレスを用いて識別することによって、識別するステップであって、前記スライスは、前記C T Bラスタスキャン順におけるC T B列の第1のシンタックス要素を含む、ステップと、

前記C T Bラスタスキャン順を用いて復号を実行するステップと、所定のC T Bの解析

10

20

30

40

50

を実行する際に、C T B単位で測定した、前記所定のC T BのC T Bアドレスモジュロ前記画像の幅の整数倍が0であるか2であるかをチェックし、C T B単位で測定した、前記所定のC T Bの前記C T Bアドレスモジュロ前記画像の幅の整数倍が2であれば、コンテキスト表内のC A B A Cコンテキスト変数の現在の状態を記憶し、C T B単位で測定した、前記所定のC T Bの前記C T Bアドレスモジュロ前記画像の幅の整数倍が0であれば、前記コンテキスト表を使って前記C A B A Cコンテキスト変数を初期化するステップと、

前記スライス順序に従って、千鳥状に前記W P Pサブストリームの前記復号化を連続的に開始するとともに前記W P Pサブストリームを並列復号化するステップをさらに含む、方法。

【請求項6】

C T B（符号化ツリーブロック）のアレイに再分割された画像（10）を、前記画像（10）がパーティション分割されているスライス（14）単位で、W P P処理を用いて、データストリーム（12）に符号化するための方法であって、前記方法は、スライス順序（16）に従って前記スライス（14）を前記データストリーム（12）に符号化するステップと、それぞれのスライス（14）に対し、前記画像（10）内における前記それぞれのスライスの符号化開始点があるC T Bを、C T Bラスタスキャン順でアドレス指定するスライスアドレス（400）を提供するステップ、および、

それぞれのW P Pサブストリームについて、前記各W P Pサブストリームのスライス順序において最初のスライスがC T B列のC T Bラスタスキャン順において最初のシンタックス要素を含むように、前記スライスをW P Pサブストリームにグループ化するステップ、および

前記データストリームが前記C T Bラスタスキャン順を用いて復号可能であるように、前記スライス順序に従って、千鳥状に前記W P Pサブストリームの前記符号化を連続的に開始するとともに、所定のC T Bの解析を開始する際に、C T B単位で測定した、前記所定のC T BのC T Bアドレスモジュロ前記画像の幅の整数倍が0であるか2であるかをチェックし、C T B単位で測定した、前記所定のC T BのC T Bアドレスモジュロ前記画像の幅の整数倍が2であれば、コンテキスト表におけるC A B A Cコンテキスト変数の現在の状態を記憶し、C T B単位で測定した、前記所定のC T BのC T Bアドレスモジュロ前記画像の幅の整数倍が0であれば、前記コンテキスト表を使って前記C A B A Cコンテキスト変数を初期化することによって、前記W P Pサブストリームを並列符号化するステップを含む、方法。

【請求項7】

コンピュータ上で実行されるときに、請求項5または請求項6に記載の方法を実行するためのプログラムコードを有するコンピュータプログラム。

【発明の詳細な説明】

【技術分野】

【0001】

本発明は、画像の低遅延符号化に関する。

【背景技術】

【0002】

現在のH E V C設計スライスにおいて、エントロピー・スライス・（以前の軽量スライス・）タイルおよびW P P（波面並列処理）は、並列化のためのツールとして含まれる。

【0003】

ビデオ・エンコーダおよびデコーダの並列化のために、画像レベルのパーティション分割は、他のアプローチと比較していくつかの利点を有する。以前のビデオ・コーデックにおいて、H．264／A V C [1]のように、画像パーティションは、符号化効率に関して高いコストを有するレギュラー・スライスでのみ可能であった。スケーラブル並列H．264／A V C復号化のために、画像再構成のためのマクロブロック・レベルの並列処理およびエントロピー復号化のためのフレーム・レベルの並列処理を結合することが必要である。しかしながら、このアプローチは、画像待ち時間および高いメモリ使用量において

10

20

30

40

50

制限された低減をもたらす。これらの制限を克服するために、新しい画像パーティション・ストラテジーが、H E V Cコーデックに含まれている。現在の参照ソフトウェア・バージョン（H M - 6）は、4つの異なるアプローチ：レギュラーまたはノーマル・スライス、エントロピー・スライス、波面並列処理（W P P）サブストリームおよびタイルを含む。典型的に、それらの画像パーティションは、1セットの最大の符号化ユニット（L C U）、または、同義語において、H E V Cまたはそれらのサブセットにおいて定義されるように、符号化ツリー・ユニット（C T U）を含む。

【0004】

図1は、画像においてL C Uまたはマクロブロックの行902ごとにレギュラー・スライス900に例示的に配置される画像898を示す。（H. 264 [1]において定義されるような）レギュラーまたはノーマル・スライスは、それらがエントロピー復号化および予測依存性を壊すように、最大の符号化ペナルティを有する。

10

エントロピー・スライスは、スライスのように、エントロピー復号化依存性を壊すが、予測（およびフィルタリング）がスライス境界を越えることを可能にする。

【0005】

W P Pにおいて、画像パーティションは、行インターリーブされ、さらに、エントロピー復号化および予測の両方は、他のパーティションにおいてブロックからデータを用いるために可能にされる。このように、符号化損失が最小化され、それと同時に、波面並列処理を利用することができる。しかしながら、インターリーブングは、従来のパーティションが復号化するために次のパーティションを必要とするように、ビットストリーム因果関係を妨害する。

20

【0006】

図2は、水平にパーティション分割するタイル906の2つの行904a、904bに分割される画像898を例示的に示す。タイルは、画像898をタイル列912a、b、cおよび行904a、bにパーティション分割する、水平908および垂直境界910を定義する。レギュラー・スライス900と同様に、タイル906は、エントロピー復号化および予測依存性を壊すが、タイルごとにヘッダを必要としない。

【0007】

これらの技術のそれぞれのために、パーティションの数を、エンコーダによって自由に選択することができる。一般に、より多くのパーティションを有することは、より高い圧縮損失をもたらす。しかしながら、W P Pにおいて、損失伝搬は、それほど高くなく、したがって、画像パーティションの数も、行ごとに1に固定することができる。これは、いくつかの利点をもたらす。第一に、W P Pのために、ビットストリーム因果関係が保証される。第二に、デコーダ実施は、一定量の並列処理が利用できると仮定することができ、それは、解像度と共に増加する。さらに、最後に、コンテキスト選択および予測依存性のいずれも、波面順序において復号化するときに壊される必要はなく、比較的低符号化損失をもたらす。

30

【0008】

しかしながら、今まで、変換概念においてすべての並列符号化は、低遅延を保つことと共に高い圧縮効率の達成を提供することができない。これは、W P P概念についても同様である。スライスは、符号化パイプラインにおいて、トランスポートーションの最小ユニットであり、さらに、いくつかのW P Pサブストリームは、まだ連続的に転送されなければならない。

40

【先行技術文献】

【非特許文献】

【0009】

【非特許文献1】 [1] Thomas Wiegand, Gary J. Sullivan, Gisle Bjontegaard, Ajay Luthra, "Overview of the H.264/AVC Video Coding Standard", IEEE Trans. Circuits Syst. Video Technol., vol. 13, N7, July 2003.

【非特許文献2】 [2] JCT-VC, "High-Efficiency Video Coding (HEVC) text spe

50

cification Working Draft 6", JCTVC-H1003, February 2012.

【非特許文献 3】 [3] ISO/IEC 13818-1: MPEG-2 Systems specification.

【発明の概要】

【発明が解決しようとする課題】

【 0 0 1 0 】

したがって、本発明の目的は、増加した効率で、例えばエンドツーエンド遅延をさらに低減してまたは費やされる符号化オーバーヘッドを低減することによって符号化効率を高めて、例えば波面並列処理に従って並列復号化を可能にする、画像符号化概念を提供することである。

【課題を解決するための手段】

10

【 0 0 1 1 】

この目的は、独立した請求項の主題によって達成される。

【 0 0 1 2 】

本発明の基本的な知見は、スライスが、それぞれのスライスの外側の画像の領域から完全に独立して、または、エントロピー符号化が関する限り少なくともそれぞれのスライスの外側の領域から独立して、符号化され／復号化される、通常のスライスの概念が、異なるモードのスライス、すなわち、例えば、スライス境界を越える相互依存性を可能にする依存スライスと呼ばれるものと、可能にしないノーマル・スライスと呼ばれる他のものを支持して、断念される場合、例えば波面並列処理などの並列処理概念が、低減されたエンドツーエンド遅延で実現され得ることである。

20

【 0 0 1 3 】

第 1 のものと結合され得りまたは個々に用いられ得る本発明のさらなる基本的な知見は、スライスの開始構文部分が W P P エントリ・ポイントの位置を決めるために用いられる場合、W P P 処理概念がより効率的に作られ得ることである。

【 0 0 1 4 】

本願の好適な実施形態は、図に関して以下に記載され、有利な実施形態は、従属した請求項の主題である。

【図面の簡単な説明】

【 0 0 1 5 】

【図 1】図 1 は、画像において L C U またはマクロブロックの行ごとにレギュラー・スライスに例示的にパーティション分割される画像を示す。

30

【図 2】図 2 は、水平にパーティション分割されたタイルの 2 つの行に例示的に分割される画像を示す。

【図 3】図 3 は、スライスまたはネットワーク転送セグメントへの並列符号化されたパーティションの割当てを例示的に示す。

【図 4】図 4 は、最小エンドツーエンド遅延のためのタイル符号化アプローチでフレームの一般的なフラグメンテーションを示す概略図を示す。

【図 5】図 5 は、最小エンドツーエンド遅延のための W P P 符号化アプローチでフレームの例示的なフラグメンテーションを示す概略図を示す。

【図 6】図 6 は、ビデオ・サービスを用いて会話のシナリオを示す概略ブロック図を示す。

40

【図 7】図 7 は、最小エンドツーエンド遅延で一般的なサブセットを有するタイルのための符号化、伝送および復号化の可能なタイム・スケジューリングを概略的に示す。

【図 8】図 8 は、エンドツーエンド遅延を共通に達成するタイミング・スケジュールを概略的に示す。

【図 9】図 9 は、2 つのスライスにパーティション分割される 11×9 個の符号化ツリーブロックを例示的に有する画像を示す。

【図 10】図 10 は、3 つのタイルにパーティション分割される 13×8 個の符号化ツリーブロックを例示的に有する画像を示す。

【図 11 a】図 11 は、シーケンス・パラメータ・セット構文のための例を示す。

50

【図 1 1 b】図 1 1 は、シーケンス・パラメータ・セット構文のための例を示す。

【図 1 1 c】図 1 1 は、シーケンス・パラメータ・セット構文のための例を示す。

【図 1 2 a】図 1 2 は、画像パラメータ・セット構文のための例を示す。

【図 1 2 b】図 1 2 は、画像パラメータ・セット構文のための例を示す。

【図 1 3 a】図 1 3 は、スライス・ヘッダ構文のための例を示す。

【図 1 3 b】図 1 3 は、スライス・ヘッダ構文のための例を示す。

【図 1 3 c】図 1 3 は、スライス・ヘッダ構文のための例を示す。

【図 1 4】図 1 4 は、レギュラー・スライスへの W P P 処理のための、および、依存スライスへの低遅延処理のための、画像のパーティションを例示する。

【図 1 5】図 1 5 は、画像パラメータ・セット構文内の部分のための例を示す。

10

【図 1 6】図 1 6 は、可能なスライス・ヘッダ構文を示す。

【図 1 7】図 1 7 は、ノーマル・スライス（および依存スライス）のための符号化相互依存性を概略的に示す。

【図 1 8】図 1 8 は、タイルの低遅延転送のための符号化（依存スライスを用いた波面並列処理）を比較する概略図を示す。

【図 1 9】図 1 9 は、図 1 8 の右側に示すように依存スライスを用いた波面並列処理を用いるときに、パイプライン低遅延伝送で例示的な W P P 符号化を示すタイミング・スケジュールを示す。

【図 2 0】図 2 0 は、アンカーとしてレギュラー・スライスを用いることによってロバスト性改善を示す概略図を示す。

20

【図 2 1】図 2 1 は、スライス・ヘッダ構文のための別の実施形態を示す。

【図 2 2】図 2 2 は、画像パラメータ・セット構文のための別の実施形態を示す。

【図 2 3】図 2 3 は、左側画像境界で開始する場合に依存スライスのためのシンボル確率初期化プロセスを示す概略図を示す。

【図 2 4】図 2 4 は、デコーダの概略図を示す。

【図 2 5】図 2 5 は、符号化ブロックおよびスライスへの画像のパーティション分割を概略的に示すとともに、デコーダのブロック図を概略的に示す。

【図 2 6】図 2 6 は、エンコーダのブロック図を概略的に示す。

【図 2 7】図 2 7 は、ここではスライス・セグメントと呼ばれる、ノーマルおよび依存スライスにパーティション分割される画像を概略的に示す。

30

【図 2 8】図 2 8 a および図 2 8 b は、ここでは一方でスライス・セグメントと呼ばれ、他方でタイルと呼ばれる、ノーマルおよび依存スライスにパーティション分割される画像を概略的に示す。

【図 2 9】図 2 9 は、依存スライスを用いたコンテキスト初期化プロセスを示すフロー図を示す。

【図 3 0】図 3 0 は、依存スライスを用いるためのコンテキスト記憶プロセスを示すフロー図を示す。

【図 3 1】図 3 1 は、W P P エントリ・ポイントの信号伝達の異なる可能性を概略的に示す。

【発明を実施するための形態】

40

【0 0 1 6】

以下において、説明は、それぞれ、並列画像処理および低遅延符号化を可能にするための現在の概念の説明から開始する。両方の能力を有することを望むときに生じる問題が概説される。特に、以下の説明から分かるように、今までのところ何とか教示されているような W P P サブストリーム概念は、それを 1 つのスライスにグループ化することによって W P P サブストリームを伝達する必要性のために低遅延を有することの望みと矛盾する。以下の実施形態は、スライス概念を広げることによって、すなわち、後に依存スライスと呼ばれる、別のタイプのスライスを導入することによって、さらに少ない遅延を必要とするアプリケーションに適用できる、例えば W P P 概念などの並列処理概念をレンダーする。

50

【0017】

キャプチャリングからディスプレイへのエンドツーエンドビデオ遅延の最小化は、例えばビデオ会議などのアプリケーションなどにおいて主要な目的の1つである。

【0018】

デジタルビデオ伝送のための信号処理チェーンは、カメラ、キャプチャリングデバイス、エンコーダ、カプセル化、伝送、デマルチプレクサ、デコーダ、レンダラーおよびディスプレイからなる。これらのステージのそれぞれは、後のステージへのそのシリアル伝送の前にイメージ・データをバッファリングすることによってエンドツーエンド遅延に貢献する。

【0019】

いくつかのアプリケーションは、そのような遅延の最小化を必要とし、例えば、操作される物体を直接的に見ることなしに、危険な領域における物体の遠隔操作、または最小侵襲手術である。短い遅延でさえも、適切な操作の重大な問題をもたらすことがありまたは重大なミスをもたらすこともある。

【0020】

多くの場合、全ビデオフレームは、例えばイントラ・フレーム処理を可能にするために、処理ステージ内でバッファリングされる。いくつかのステージは、次のステージに送られるパケットを形成するためにデータを集める。一般に、ローカル処理の要件から生じる遅延のための下方境界がある。これは、以下にさらに詳細に個々のステージごとに分析される。

【0021】

カメラ内の処理は、最小遅延がフレーム・レートによって制限されるセンサの積分時間およびハードウェア・メーカーによるいくつかの設計上の選択によって与えられるように、イントラ・フレーム信号処理を必ずしも必要とするというわけではない。カメラ出力は、通常、左上隅において処理を開始し、右上隅に移動し、さらに、ラインごとに右下隅に続く、スキャン順序に典型的に関連している。結果的に、すべてのデータがセンサからカメラ出力に転送されるまで約1フレーム期間を要する。

【0022】

キャプチャリングデバイスは、受信の直後にカメラ・データを送ることができるが、それは、典型的に、いくつかのデータをバッファリングし、さらに、メモリまたはストレージへのデータ・アクセスを最適化するためにバーストを生成する。さらに、カメラ／キャプチャおよびコンピュータのメモリ間の接続は、さらなる処理（符号化）のためのメモリに、キャプチャされたイメージ・データを送るためのビットレートを典型的に制限している。典型的に、カメラは、USB 2.0を介して、または、すぐにUSB 3.0で接続され、それは、エンコーダにイメージ・データの部分的な転送を常に含む。これは、極端な低遅延シナリオにおいてエンコーダ側における並列化能力を制限し、すなわち、エンコーダは、データが例えばイメージの上から下までラスタ・スキャン順序においてカメラから利用できるようになるときに、できるだけ早く符号化を開始しようとする。

【0023】

エンコーダにおいて、特定のビデオ忠実度のために必要とされるデータ・レートに関して、処理遅延の低減のために、符号化効率をトレードオフすることを可能にするいくらかの自由度がある。

【0024】

エンコーダは、後に符号化されるイメージを予測するためにすでに送られたデータを用いる。一般に、実際のイメージおよび予測間の差を、予測なしに必要とされるよりも少ないビットで符号化することができる。この予測値は、デコーダで利用できるようにする必要があり、このように、予測は、同じイメージ（イントラ・フレーム予測）の前に復号化された部分、または、前に処理された他のイメージ（インター・フレーム予測）に基づく。プレHEVCビデオ符号化標準は、同じラインの上方にまたは上にあるが左にあるイメージの部分だけを用い、それは、イントラ・フレーム予測、運動ベクトル予測およびエン

10

20

30

40

50

トロピー符号化 (C A B A C) のために、前に符号化されている。

【0025】

予測構造の最適化に加えて、並列処理の影響を考慮することができる。並列処理は、独立して処理することができる画像領域の識別を必要とする。実用的な理由から、例えば水平または垂直矩形などの連続した領域が選択され、それは、しばしば「タイル」と呼ばれる。低遅延制約の場合、それらの領域は、できるだけ早く、キャプチャからメモリに入力するデータの並列化された符号化を可能にすべきである。ラスト・スキャン・メモリ転送を仮定すれば、直ちに符号化を開始するために、生データの垂直パーティションが意味をなす。画像を垂直パーティションに分割する(下図を参照)、そのようなタイルの内部において、イントラ予測、運動ベクトル予測およびエントロピー符号化 (C A B A C) は、
10 相当な符号化効率をもたらすことができる。遅延を最小化するために、上から開始する、画像の部分だけは、エンコーダのフレーム・メモリに転送され、さらに、並列処理は、垂直タイルにおいて開始されるべきである。

【0026】

並列処理を可能にする別の方法は、タイルと比較されるレギュラー・スライス内でWPPを用いることであり、タイルの「行」は、単一のスライスに含まれている。スライス内のデータは、スライス内で、WPPサブストリームを用いて並列符号化され得る。スライス900およびタイル/WPPサブストリーム914への画像分離が、例の図3/1の形で示される。
20

【0027】

このように、図3は、例えばスライスまたはネットワーク転送セグメント(単一のネットワーク・パケットまたは複数のネットワーク900パケット)への906または914などの並列符号化されたパーティションの割当てを示す。

【0028】

ネットワーク抽象化層(NAL)ユニットへの符号化されたデータのカプセル化は、適用できる場合、H.264またはHEVCにおいて定義されるように、伝送の前にまたは符号化プロセスの間に、いくつかのヘッダをデータ・ブロックに加え、それは、それぞれのブロックの識別およびブロックの並び換えを可能にする。標準の場合、付加的な信号伝達は、符号化要素の順序が常に復号化順序であるので、必要とされなく、それは、タイルまたは一般的な符号化フラグメントの位置の暗黙の割当てが与えられることである。
30

【0029】

並列処理が低遅延並列転送のための付加的なトランスポート層で考慮される場合、すなわち、トランスポート層は、それらが符号化されるように図4に示すようなフラグメントを送ることを意味する、低遅延伝送を可能するためにタイルのための画像パーティションを並び換えることができる。それらのフラグメントは、完全に符号化されていないスライスであってもよく、それらは、スライスのサブセットであってもよく、または、依存スライスに含まれてもよい。

【0030】

付加的なフラグメントを作る場合、ヘッダ情報が一定数のバイトを加えていることによる大きいデータ・ブロックで最も高い効率と、並列エンコーダの大きいデータ・ブロックが伝送の前にバッファリングされる必要があることによる遅延との間のトレードオフがある。垂直タイル906の符号化された表現が、フラグメントが完全に符号化されるとすぐに送信される多数のフラグメント916において分離される場合、全体の遅延を低減することができる。それぞれのフラグメントのサイズを、例えばマクロブロック、LCUなどのように、固定されたイメージ領域に関して、または図4に示すような最大のデータに関して、決定することができる。
40

【0031】

このように、図4は、最小エンドツーエンド遅延のためのタイル符号化アプローチでフレームの一般的なフラグメンテーションを示す。

【0032】

同様に、図5は、最小エンドツーエンド遅延のためのWPP符号化アプローチでフレームのフラグメンテーションを示す。

【0033】

伝送は、例えば、付加的なブロック指向の処理が適用される場合、例えば方向誤り訂正が伝送のロバスト性を増加して符号化するように、さらなる遅延を加えることができる。また、ネットワーク・インフラストラクチャ（ルータなど）または物理結合は、遅延を加えることができ、これは、接続のための待ち時間として典型的に知られている。待ち時間に加えて、伝送ビットレートは、ビデオ・サービスを用いる図6に示すような会話において、データをパーティaからパーティbに転送するための時間（遅延）を決定する。

【0034】

符号化されたデータ・ブロックが順序から外れて送信される場合、遅延の並び換えが考慮されなければならない。復号化は、データ・ユニットが到着するとすぐに開始することができ、これの前に復号化されなければならない他のデータ・ユニットが利用できると仮定される。

【0035】

タイルの場合、タイル間に依存性がなく、そのため、タイルを直ちに復号化することができる。フラグメントが、例えば図4に示すようなフラグメントごとに別々のスライスなどように、タイルで作られている場合、フラグメントは、それらがそれぞれ符号化され、それらの含まれたLCUまたはCUが符号化されているとすぐに、直接的に転送され得る。

【0036】

レンダラーは、並列復号化エンジンの出力をアSEMBルし、さらに、結合された画像をラインごとにディスプレイに送る。

【0037】

ディスプレイは、いかなる遅延も必ずしも加えるというわけではないが、実際には、イメージ・データが実際に表示される前に、いくつかのイントラ・フレーム処理を行うことができる。これは、ハードウェア・メーカーによる設計上の選択次第である。

【0038】

要約すると、我々は、最小エンドツーエンド遅延を達成するために、ステージ符号化、カプセル化、伝送および復号化に影響を与えることができる。我々が並列処理、タイルおよびタイル内でのフラグメンテーションを用いる場合、図8に示すようなこれらのステージのそれぞれで約1フレーム遅延を加える一般に用いられる処理チェーンと比較して、全体の遅延を、図7に示すように著しく低減することができる。

【0039】

特に、図7は、最小エンドツーエンド遅延で一般的なサブセットを有するタイルのための符号化、伝送および復号化を示す一方、図8は、共通に達成されたエンドツーエンド遅延を示す。

【0040】

HEVCは、スライス・パーティション分割、タイル・パーティション分割の使用をさらに以下の方法において可能にする。

【0041】

タイル：1つの列および1つの行において同時に生じる整数個のツリーブロックであって、タイルのツリーブロック・ラスタ・スキャンにおいて連続的に順序付けられる。タイルへのそれぞれの画像の分割は、パーティション分割である。画像においてタイルは、画像のタイル・ラスタ・スキャンにおいて連続的に順序付けられる。スライスがタイルのツリーブロック・ラスタ・スキャンにおいて連続的であるツリーブロックを含むにもかかわらず、これらのツリーブロックは、画像のツリーブロック・ラスタ・スキャンにおいて必ずしも連続的であるというわけではない。

【0042】

スライス：ラスタ・スキャンにおいて連続的に順序付けられる整数個のツリーブロック

10

20

30

40

50

。スライスへのそれぞれの画像の分割は、パーティション分割である。ツリーブロック・アドレスは、（スライス・ヘッダにおいて表されるように）スライスにおいて第1のツリーブロック・アドレスから導出される。

【0043】

ラスタ・スキャン：1次元パターンにおいて第1のエントリが、左から右にスキャンされる2次元パターンの一番上の第1の行から、それぞれが左から右にスキャンされ（下がる）パターンの第2、第3などの行が同様に続くように、1次元パターンへの矩形の2次元パターンのマッピング。

【0044】

ツリーブロック：ルマ（luma）・サンプルの $N \times N$ ブロックおよび3つのサンプル・アレイを有する画像のクロマ（chroma）・サンプルの2つの対応するブロック、または、モノクロ画像または3つの別々の色平面を用いて符号化される画像のサンプルの $N \times N$ ブロック。ツリーブロックへのスライスの分割は、パーティション分割である。

【0045】

パーティション分割：セットのそれぞれの要素がサブセットの正確に1つにあるようにサブセットへのセットの分割。

【0046】

クワッドツリー：親ノードを4つの子ノードに分割することができるツリー。子ノードは、4つの子ノードへの別の分割のための親ノードになることがある。

【0047】

以下において、画像、スライスおよびタイルの空間再分割が説明される。特に、以下の説明は、どのように画像がスライス、タイルおよび符号化ツリーブロックにパーティション分割されるかについて指定する。画像は、スライスおよびタイルに分割される。スライスは、一連の符号化ツリーブロックである。同様に、タイルは、一連の符号化ツリーブロックである。

【0048】

サンプルは、符号化ツリーブロックを単位にして処理される。幅および高さの両方においてサンプルにおけるツリーブロックごとのルマ（luma）・アレイ・サイズは、CtbSizeである。符号化ツリーブロックごとのクロマ（chroma）・アレイの幅および高さは、それぞれ、CtbWidthCおよびCtbHeightCである。例えば、画像は、次の図に示すように2つのスライスに分割され得る。別の例として、画像は、第2の以下の図に示すように3つのタイルに分割され得る。

【0049】

スライスとは異なり、タイルは、常に矩形であり、さらに、常に符号化ツリーブロック・ラスタ・スキャンにおいて整数個の符号化ツリーブロックを含む。タイルは、1つよりも多いスライスに含まれる符号化ツリーブロックからなり得る。同様に、スライスは、1つよりも多いタイルに含まれる符号化ツリーブロックからなり得る。

【0050】

図9は、2つのスライス900a、bにパーティション分割される 11×9 の符号化ツリーブロック918を有する画像898を示す。

【0051】

図10は、3つのタイルにパーティション分割される 13×8 の符号化ツリーブロック918を有する画像を示す。

【0052】

それぞれの符号化898のツリーブロック918は、イントラまたはインター予測のためのおよび変換符号化のための、ブロック・サイズを識別するために信号伝達するパーティションが割り当てられる。パーティション分割は、再帰的なクワッドツリー・パーティション分割である。クワッドツリーのルートは、符号化ツリーブロックと関連付けられる。クワッドツリーは、リーフに達するまで分割され、それは、符号化ブロックと呼ばれる。符号化ブロックは、2つのツリーのルート・ノード、予測ツリーおよび変換ツリーであ

10

20

30

40

50

る。

【0053】

予測ツリーは、予測ブロックの位置およびサイズを指定する。予測ブロックおよび関連した予測データは、予測ユニットと呼ばれる。

【0054】

図11は、例示的なシーケンス・パラメータ・セットRBS P構文を示す。

【0055】

変換ツリーは、変換ブロックの位置およびサイズを指定する。変換ブロックおよび関連した変換データは、変換ユニットと呼ばれる。

【0056】

ルマ (luma) およびクロマ (chroma) のための分割情報は、予測ツリーのと同一であり、さらに、変換ツリーのと同一であってもよくまたは同一でなくてもよい。

【0057】

符号化ブロック、関連した符号化データ、関連した予測および変換ユニットは、一緒に符号化ユニットを形成する。

【0058】

タイル・スキャン順序への符号化ツリーブロック・ラスタ順序において符号化ツリーブロック・アドレスの変換のためのプロセスは、以下のようになり得る：

このプロセスの出力は、

— 0からPicHeightInCtbs*PicWidthInCtbs-1の範囲においてctbAddrRSを有する、アレイCtbAddrTS[ctbAddrRS]

— 0からPicHeightInCtbs*PicWidthInCtbs-1の範囲においてctbAddrTSを有する、アレイTileId[ctbAddrTS]である。

アレイCtbAddrTS[]は、以下のように導出される：

```
for( ctbAddrRS = 0; ctbAddrRS < PicHeightInCtbs * PicWidthInCtbs; ctbAddrRS++) {
    tbX = ctbAddrRS % PicWidthInCtbs
    tbY = ctbAddrRS / PicWidthInCtbs
    for( j = 0; j <= num_tile_columns_minus1; j++)
        if( tbX < ColBd[ j + 1 ] )
            tileX = j
    for( i = 0; i <= num_tile_rows_minus1; i++)
        if( tbY < RowBd[ i + 1 ] )
            tileY = i
    CtbAddrTS[ ctbAddrRS ] = ctbAddrRS - tbX
    for( i = 0; i < tileX; i++)
        ctbAddrTS += RowHeight[ tileY ] * ColumnWidth[ i ]
    CtbAddrTS[ ctbAddrRS ] += ( tbY - RowBd[ tileY ] ) * ColumnWidth[ tileY ] + tbX - ColBd[ tileX ]
}
```

【0059】

アレイTileId[]は、以下のように導出される：

```
for( j = 0; tileId = 0; j <= num_tile_columns_minus1; j++)
    for( i = 0; i <= num_tile_rows_minus1; i++, tileId++)
        for( y = RowBd[ j ]; y < RowBd[ j + 1 ]; y++)
            for( x = ColBd[ i ]; x < ColBd[ i + 1 ]; x++)
                TileId[ CtbAddrTS[ y*PicWidthInCtbs + x ] ] = tileId
```

【0060】

対応する、例示的な構文は、図11、図12および図13に示され、図12は、例示的な画像パラメータ・セットRBS P構文を有する。図13は、例示的なスライス・ヘッダ構文を示す。

【0061】

構文例において、以下のセマンティックが適用され得る：

【0062】

1に等しい`entropy_slice_flag`は、存在しないスライス・ヘッダ構文要素の値が進行スライスにおいてスライス・ヘッダ構文要素の値に等しいと推測されることを指定し、進行スライスは、位置を有する符号化ツリーブロックを含むスライスとして定義される（`SliceCtbAddrRS-1`）。`entropy_slice_flag`は、`SliceCtbAddrRS`が0に等しいときに0に等しい。

【0063】

0に等しい`tiles_or_entropy_coding_sync_idc`は、符号化されたビデオ・シーケンスにおけるそれぞれの画像において1つのタイルだけがあることを指定し、さらに、コンテキスト変数のための特定の同期化プロセスは、符号化ツリーブロックの行の第1の符号化ツリーブロックを復号化する前に呼び出されない。

10

【0064】

1に等しい`tiles_or_entropy_coding_sync_idc`は、符号化されたビデオ・シーケンスにおけるそれぞれの画像において1つよりも多いタイルがあってもよいことを指定し、さらに、コンテキスト変数のための特定の同期化プロセスは、符号化ツリーブロックの行の第1の符号化ツリーブロックを復号化する前に呼び出されない。

【0065】

2に等しい`tiles_or_entropy_coding_sync_idc`は、符号化されたビデオ・シーケンスにおけるそれぞれの画像において1つのタイルだけがあることを指定し、コンテキスト変数のための特定の同期化プロセスは、符号化ツリーブロックの行の第1の符号化ツリーブロックを復号化する前に呼び出され、さらに、コンテキスト変数のための特定の暗記プロセスは、符号化ツリーブロックの行の2つの符号化ツリーブロックを復号化した後に呼び出される。

20

【0066】

`tiles_or_entropy_coding_sync_idc`の値は、0から2の範囲にある。

【0067】

`num_tile_columns_minus1+1`は、画像をパーティション分割するタイル列の数を指定する。

30

【0068】

`num_tile_rows_minus1+1`は、画像をパーティション分割するタイル行の数を指定する。`num_tile_columns_minus1`が0に等しいときに、`num_tile_rows_minus1`は、0に等しくない。

【0069】

以下の状況の一方または両方は、スライスおよびタイルごとに満たされる：

- － スライスにおいてすべての符号化されたブロックは、同じタイルに属する。
- － タイルにおいてすべての符号化されたブロックは、同じスライスに属する。

【0070】

注意－ 同じ画像内において、複数のタイルを含むスライスおよび複数のスライスを含むタイルの両方があってもよい。

40

【0071】

1に等しい`uniform_spacing_flag`は、列境界および同様に行境界が画像にわたって一様に分布されることを指定する。0に等しい`uniform_spacing_flag`は、列境界および同様に行境界が画像にわたって一様に分布されないが、構文要素`column_width[i]`および`row_height[i]`を用いて明確に信号伝達されることを指定する。

【0072】

`column_width[i]`は、符号化ツリーブロックを単位にして*i*番目のタイ

50

ル列の幅を指定する。

【0073】

`row_height[i]` は、符号化ツリーブロックを単位にして i 番目のタイル行の高さを指定する。符号化ツリーブロックを単位にして i 番目のタイル列の幅を指定する `ColumnWidth[i]` の値、およびルマ (luma) ・サンプルを単位にして i 番目のタイル列の幅を指定する `ColumnWidthInLumaSamples[i]` の値は、以下のように導出される：

```
for( i = 0; i <= num_tile_columns_minus1; i++ ) {
    if( uniform_spacing_flag )
        ColumnWidth[ i ] = ( ( i + 1 ) * PicWidthInCtbs ) / ( num_tile_columns_minus1 + 1 ) -
                            ( i * PicWidthInCtbs ) / ( num_tile_columns_minus1 + 1 )
    else
        ColumnWidth[ i ] = column_width[ i ]
        ColumnWidthInLumaSamples[ i ] = ColumnWidth[ i ] << Log2CtbSize
}
```

【0074】

符号化ツリーブロックを単位にして i 番目のタイル行の高さを指定する `RowHeight[i]` の値は、以下のように導出される：

```
for( i = 0; i <= num_tile_rows_minus1; i++ )
    if( uniform_spacing_flag )
        RowHeight[ i ] = ( ( i + 1 ) * PicHeightInCtbs ) / ( num_tile_rows_minus1 + 1 ) -
                        ( i * PicHeightInCtbs ) / ( num_tile_rows_minus1 + 1 )
    else
        RowHeight[ i ] = row_height[ i ]
```

【0075】

符号化ツリーブロックを単位にして i 番目のタイル列の左側列境界の位置を指定する `ColBd[i]` の値は、以下のように導出される：

```
for( ColBd[ 0 ] = 0, i = 0; i <= num_tile_columns_minus1; i++ )
    ColBd[ i + 1 ] = ColBd[ i ] + ColumnWidth[ i ]
```

【0076】

符号化ツリーブロックを単位にして i 番目のタイル行の一番上の行境界の位置を指定する `RowBd[i]` の値は、以下のように導出される：

```
for( RowBd[ 0 ] = 0, i = 0; i <= num_tile_rows_minus1; i++ )
    RowBd[ i + 1 ] = RowBd[ i ] + RowHeight[ i ]
```

【0077】

`num_substreams_minus1 + 1` は、`tiles_or_entropy_coding_sync_idc` が 2 に等しいときに、スライスに含まれるサブセットの最大数を指定する。存在しないときに、`num_substreams_minus1` の値は、0 に等しいと推測される。

【0078】

`num_entry_point_offsets` は、スライス・ヘッダにおいて `entry_point_offset[i]` 構文要素の数を指定する。`tiles_or_entropy_coding_sync_idc` が 1 に等しいときに、`num_entry_point_offsets` の値は、0 から $(\text{num_tile_columns_minus1} + 1) * (\text{num_tile_rows_minus1} + 1) - 1$ の範囲にある。`tiles_or_entropy_coding_sync_idc` が 2 に等しいときに、`num_entry_point_offsets` の値は、0 から `num_`

`streams_minus1`の範囲にある。存在しないときに、`num_entry_point_offsets`の値は、0に等しいと推測される。

【0079】

`offset_len_minus1+1`は、ビットにおいて、`entry_point_offset[i]` 構文要素の長さを指定する。

【0080】

`entry_point_offset[i]` は、バイトにおいて、*i* 番目のエントリ・ポイント・オフセットを指定し、さらに、`offset_len_minus1+1`のビットによって表される。符号化されたスライスNALユニットは、0から`num_entry_point_offsets`の範囲のサブセット・インデックス値を有する、`num_entry_point_offsets+1`のサブセットからなる。サブセット0は、0から符号化されたスライスNALユニットの`entry_point_offset[0]-1`のバイトからなり、1から`num_entry_point_offsets-1`の範囲において*k*を有するサブセット*k*は、符号化されたスライスNALユニットの`entry_point_offset[k-1]` から`entry_point_offset[k]+entry_point_offset[k-1]-1`のバイトからなり、さらに、(`num_entry_point_offsets`に等しいサブセット・インデックスを有する)最後のサブセットは、符号化されたスライスNALユニットの残りのバイトからなる。

10

【0081】

注意— NALユニット・ヘッダおよび符号化されたスライスNALユニットのスライス・ヘッダは、サブセット0に常に含まれる。

20

【0082】

`tiles_or_entropy_coding_sync_idc`が1に等しく、さらに、`num_entry_point_offsets`が0よりも大きいときに、それぞれのサブセットは、1または複数の完全なタイルのすべての符号化されたビットを含み、さらに、サブセットの数は、スライスにおいてタイルの数以下である。

`tiles_or_entropy_coding_sync_idc`が2に等しく、さらに、`num_entry_point_offsets`が0よりも大きいときに、サブセット*k*は、すべての可能な*k*値のそれぞれのために、現在のビットストリーム・ポイント*k*のための初期化プロセスの間に用いられるすべてのビットを含む。

30

【0083】

スライス・データ・セマンティックに関して、以下が適用され得る。

【0084】

0に等しい`end_of_slice_flag`は、別のマクロブロックがスライスにおいて続いていることを指定する。1に等しい`end_of_slice_flag`は、スライスの最後を指定し、さらに、さらなるマクロブロックは、続かない。

【0085】

`entry_point_marker_two_3bytes`は、0x000002に等しい3バイトの固定値シーケンスである。この構文要素は、エントリ・マーカー・プレフィックスと呼ばれる。

40

【0086】

`tile_idx_minus_1`は、ラスタ・スキャン順序においてTileIDを指定する。画像において第1のタイルは、0のTileIDを有する。`tile_idx_minus_1`の値は、0から(`num_tile_columns_minus1+1`) * (`num_tile_rows_minus1+1`) - 1の範囲にある。

【0087】

スライス・データのためのCABAC解析プロセスは、以下のようになり得る：

【0088】

このプロセスは、記述子`ae(v)`を有する構文要素を解析するときに呼び出される。

50

【0089】

このプロセスへの入力、構文要素の値および従来の解析された構文要素の値のための要請である。

【0090】

このプロセスの出力は、構文要素の値である。

【0091】

スライスのスライス・データの解析を開始するときに、CABAC解析プロセスの初期化プロセスが呼び出される。`tiles_or_entropy_coding_sync_idc`が2に等しく、さらに、`num_substreams_minus1`が0よりも大きいときに、後の現在のビットストリーム・ポインタ導出のために用いるためにビットストリーム・ポインタ・テーブルを指定する`num_substreams_minus1+1`のエントリを有するマッピング・テーブル`BitStreamTable`が以下のように導出される。

— `BitStreamTable[0]`は、ビットストリーム・ポインタを含むために初期化される。

— 0よりも大きくさらに`num_substreams_minus1+1`よりも小さいすべてのインデックス*i*のために、`BitStreamTable[i]`は、`BitStreamTable[i-1]`の後に`entry_point_offset[i]`バイトへのビットストリーム・ポインタを含む。

現在のビットストリーム・ポインタは、`BitStreamTable[0]`にセットされる。

【0092】

空間隣接ブロックTを含む符号化ツリーブロックの最小符号化ブロック・アドレス、`ctbMinCbAddrT`は、例えば、以下のように、現在の符号化ツリーブロックの左上のルマ(luma)・サンプルの位置(`x0`、`y0`)を用いて導出される。

$$x = x0 + 2 \ll \text{Log2CtbSize} - 1$$

$$y = y0 - 1$$

$$\text{ctbMinCbAddrT} = \text{MinCbAddrZS}[x \gg \text{Log2MinCbSize}][y \gg \text{Log2MinCbSize}]$$

【0093】

変数`availableFlagT`は、入力として`ctbMinCbAddrT`を有する適切な符号化ブロック利用可能性導出プロセスを呼び出すことによって得られる。

符号化ツリーの解析を開始し、さらに、`tiles_or_entropy_coding_sync_idc`が2に等しく、さらに、`num_substreams_minus1`が0よりも大きいときに、以下が適用される。

— `CtbAddrRS % PicWidthInCtbs`が0に等しい場合、以下が適用される。

— `availableFlagT`が1に等しいときに、CABAC解析プロセスの同期化プロセスが、下位条項「コンテキスト変数のための同期化プロセス」において指定されるように呼び出される。

— 終了の前のバイナリ決定のための復号化プロセスが呼び出され、その後に算術復号化エンジンのための初期化プロセスが続く。

— 現在のビットストリーム・ポインタは、以下のように導出されるインデックス*i*を有する`BitStreamTable[i]`を示すためにセットされる。

$$i = (\text{CtbAddrRS} / \text{PicWidthInCtbs}) \% (\text{num_substreams_minus1} + 1)$$

— そうでなければ、`CtbAddrRS % PicWidthInCtbs`が2に等しい場合、CABAC解析プロセスの暗記プロセスが、下位条項「コンテキスト変数のための暗記プロセス」において指定されるように呼び出される。

【0094】

初期化プロセスは、以下のようになり得る：

【0095】

このプロセスの出力は、初期化されたC A B A C内部変数である。

【0096】

その特別なプロセスは、スライスのスライス・データの解析を開始するときに、または、符号化ツリーのデータの解析を開始しさらに符号化ツリーがタイルにおいて第1の符号化ツリーであるときに、呼び出される。

【0097】

コンテキスト変数のための暗記プロセスは、以下のようになり得る：

【0098】

このプロセスの入力は、c t x I d xによってインデックスが付けられるC A B A Cコンテキスト変数である。

【0099】

このプロセスの出力は、スライスのエンド・フラグを除いて構文要素に割り当てられるコンテキスト変数の初期化プロセスにおいて用いられる変数mおよびnの値を含む変数T a b l e S t a t e S y n cおよびT a b l e M P S S y n cである。

コンテキスト変数ごとに、テーブルT a b l e S t a t e S y n cおよびT a b l e M P S S y n cの対応するエントリnおよびmは、対応するp S t a t e I d xおよびv a l M P Sに初期化される。

【0100】

コンテキスト変数のための同期化プロセスは、以下ようになり得る：

【0101】

このプロセスの入力は、スライスのエンド・フラグを除いて構文要素に割り当てられるコンテキスト変数の暗記プロセスにおいて用いられる変数nおよびmの値を含む変数T a b l e S t a t e S y n cおよびT a b l e M P S S y n cである。

【0102】

このプロセスの出力は、c t x I d xによってインデックスが付けられるC A B A Cコンテキスト変数である。

【0103】

コンテキスト変数ごとに、対応するコンテキスト変数p S t a t e I d xおよびv a l M P Sは、テーブルT a b l e S t a t e S y n cおよびT a b l e M P S S y n cの対応するエントリnおよびmに初期化される。

【0104】

以下において、W P Pを用いる低遅延符号化および転送が説明される。特に、以下の説明は、どのように図7に記載するような低遅延転送をW P Pに適用できるかについて明らかにする。

【0105】

まず第一に、全画像の完成の前に、画像のサブセットが送られ得ることが重要である。通常、これは、すでに図5に示すように、スライスを用いて達成可能である。

【0106】

タイルと比較して遅延を低減するために、以下の図に示すように、L C Uの行ごとに単一のW P Pサブビットストリームを適用し、さらにそれらの行のそれぞれの別々の伝送を可能にする必要がある。符号化効率を高く保つために、列／サブストリームごとのスライスを用いることができない。したがって、以下で、次のセクションにおいて定義されるようないわゆる依存スライスが導入される。このスライスは、例えば、完全なH E V Cスライス・ヘッダのすべての分野ではないが、エントロピー・スライスのために用いられる分野を有する。さらに、行間でC A B A Cの破壊をオフにするためにスイッチがあってもよい。W P Pの場合、C A B A Cコンテキストの使用（図14の矢印）および行の予測は、タイル上のW P Pの符号化効率ゲインを保つために可能にされる。

【0107】

10

20

30

40

50

特に、図14は、レギュラー・スライス900（レギュラーSL）へのWPPのための、および、依存スライス（OS）920への低遅延処理のための、画像10を例示する。

【0108】

現在、今度のHEVC標準は、スライスに関してパーティション分割の2つのタイプを提供する。レギュラー（ノーマル）・スライスおよびエントロピー・スライスがある。レギュラー・スライスは、スライス境界においてフィルタ・プロセスを非ブロック化するために利用できるいくつかの依存性を除いて、完全に独立した画像パーティションである。エントロピー・スライスは、エントロピー符号化に関してだけ独立している。図14の考えは、スライシング概念を一般化することである。このように、今度のHEVC標準は、スライスの2つの一般的なタイプ：独立（レギュラー）または依存を提供すべきである。したがって、新しいタイプのスライス、依存スライスが導入される。

10

【0109】

依存スライスは、前のスライスに対して依存性を有するスライスである。依存性は、エントロピー復号化プロセスおよび／またはピクセル再構成プロセスにおいてスライス間に利用できる特定のデータである。

【0110】

図14において、依存スライスの概念が、例示的に示される。画像は、例えば、常にレギュラー・スライスから開始する。この概念において、レギュラー・スライス挙動がわずかに変更されることに注意する。典型的に、H264／AVCまたはHEVCのような標準において、レギュラー・スライスは、完全に独立したパーティションであり、さらに、フィルタ・プロセスを非ブロック化するためのいくつかのデータを除いて、復号化の後に、いかなるデータも保つ必要はない。しかし、この次の依存スライス920の処理は、上述のスライス、ここで第1の行において、レギュラー・スライス900のデータを参照することによってだけ可能である。それを確立するために、レギュラー・スライス900は、最後のCU行のデータを保つべきである。

20

このデータは、

- CABAC符号化エンジン・データ（依存スライスのエントロピー復号化プロセスが初期化され得る1つのCUのコンテキスト・モデル状態）、

- 依存CUのレギュラーCABAC復号化プロセスのためのCUのすべての復号化された構文要素、

30

- イントラおよび運動ベクトル予測のデータを含む。

【0111】

結果的に、それぞれの依存スライス920は、同じ画像においてこの次の依存スライスのためのデータを保つ、同じ手順を行うべきである。

【0112】

実際には、これらの追加的なステップは、問題であるべきでなく、なぜなら、復号化プロセスは、一般に、常に構文要素のようないくつかのデータを格納することを強いられるからである。

【0113】

40

以下のセクションにおいて、依存スライスの概念を可能にすることが必要とされるHEVC標準構文のための可能な変更が示される。

【0114】

図5は、例えば、画像パラメータ・セットRBS P構文において可能な変更を示す。

【0115】

依存スライスのための画像パラメータ・セット・セマンティックは、以下のようになり得る：

【0116】

1に等しい`dependent_slices_present_flag`は、画像が依存スライスを含むことを指定し、さらに、それぞれの（レギュラーまたは依存）スライ

50

スの復号化プロセスは、エントロピー復号化の状態と、レギュラー・スライスに続き得る依存スライスであってもよい次のスライスのためのイントラおよび運動ベクトル予測のデータとを格納すべきである。以下の依存スライスは、その格納されたデータを参照することができる。

【0117】

図16は、HEVCの現在の状況と関連する変化を有する可能なslice_header構文を示す。

【0118】

1に等しいdependent_slice_flagは、存在しないスライス・ヘッダ構文要素の値が進行（レギュラー・）スライスにおいてスライス・ヘッダ構文要素の値に等しいと推測されることを指定し、進行スライスは、位置を有する符号化ツリーブロックを含むスライスとして定義される（SliceCtbAddrRS-1）。dependent_slice_flagは、SliceCtbAddrRSが0に等しいときに0に等しい。

10

【0119】

1に等しいno_cabac_reset_flagは、前に復号化されたスライス（初期値を有しない）の保存された状態からCABAC初期化を指定する。そうでなければ、すなわち0に等しい場合、前に復号化されたスライスの、すなわち初期値を有する、いかなる状態からも独立してCABAC初期化を指定する。

20

【0120】

1に等しいlast_ctb_cabac_init_flagは、（例えば常に1に等しいタイルのために）前に復号化されたスライスの最後の符号化されたツリーブロックの保存された状態からCABAC初期化を指定する。そうでなければ（0に等しい場合）、初期化データは、現在のスライスの第1の符号化されたツリーブロックが行において第1の符号化されたツリーブロックである場合（すなわちWPPモード）、前に復号化されたスライスの最後の（隣接した）ctb-rowの第2の符号化されたツリーブロックの保存された状態から参照され、そうでなければ、CABAC初期化は、前に復号化されたスライスの最後の符号化されたツリーブロックの保存された状態から予め形成される。

【0121】

依存スライスおよび他のパーティション分割スキーム（情報）の比較は、以下に提供される。

30

【0122】

図17において、ノーマルおよび依存スライスの差が示される。

【0123】

図18に関して示されるように依存スライス（DS）においてWPPサブストリームの可能な符号化および伝送は、タイル（左）およびWPP/DS（右）の低遅延転送のための符号化に匹敵する。図18において太い連続的に描かれたクロスは、WPP行の符号化が単一のタイルの符号化と同じ時間をとると仮定すると2つの方法のための同じ時点を示す。符号化依存性のために、WPPの第1のラインだけは、すべてのタイルが符号化された後に、準備されている。しかし、依存スライス・アプローチを用いると、WPPアプローチは、それが符号化されるとすぐに第1の行を送ることを可能にする。これは、初期のWPPサブストリーム割当てと異なり、「サブストリーム」は、同じデコーダ・スレッド、すなわち同じコア/プロセッサによって復号化されるWPPであるスライスのCU行の連結としてWPPのために定義される。しかし、行ごとおよびエントロピー・スライスごとのサブストリームは、エントロピー・スライスがエントロピー符号化依存性を壊し、したがって低符号化効率を有し、すなわちWPP効率ゲインが損失する前にも可能である。

40

【0124】

加えて、両方のアプローチ間の遅延差は、図19に示すような伝送を仮定すると、実際に低くすることができる。特に、図19は、パイプライン低遅延伝送でWPP符号化を示す。

50

【0125】

図18においてWPPアプローチにおけるDS#1. 1の後の2つのCUの符号化が、第1の行SL#1の伝送よりも長くないと仮定すると、低遅延の場合においてタイルおよびWPP間の差がない。しかし、WPP/DSの符号化効率、タイル概念よりもパフォーマンスが優れている。

【0126】

WPP低遅延モードのためのロバスト性を増加するために、図20は、ロバスト性改善がアンカーとしてレギュラー・スライス(RS)を用いることによって達成されることを示す。図20に示される画像において、(レギュラー・)スライス(RS)の後に依存スライス(DS)が続く。ここで、(レギュラー・)スライスは、前のスライスへの依存性を壊すアンカーとして働き、そのため、より多くのロバスト性が(レギュラー・)スライスのそのような挿入ポイントで提供される。原理的には、これは、とにかく(レギュラー・)スライスを挿入することと変わらない。

10

【0127】

依存スライスの概念を、以下のように実施することもできる。

【0128】

ここで、図21は、可能なスライス・ヘッダ構文を示す。

【0129】

スライス・ヘッダ・セマンティックは、以下のとおりである：

【0130】

20

1に等しいdependent__slice__flagは、存在しないそれぞれのスライス・ヘッダ構文要素の値が、符号化ツリー・ブロック・アドレスがSliceCtbAddrRS-1である符号化ツリー・ブロックを含む前のスライスにおいて対応するスライス・ヘッダ構文要素の値に等しいと推測されることを指定する。存在しないときに、dependent__slice__flagの値は、0に等しいと推測される。dependent__slice__flagの値は、SliceCtbAddrRSが0に等しいときに0に等しい。

【0131】

slice__addressは、スライスが開始するスライス粒度解像度においてアドレスを指定する。slice__address構文要素の長さは、(Ciel(Log2(PicWidthInCtbs*PicHeightInCtbs))+SliceGranularity)ビットである。

30

【0132】

スライスが符号化ツリー・ブロック・ラスタ・スキャン順序において開始する符号化ツリー・ブロックを指定する、変数SliceCtbAddrRSは、以下のように導出される。

$$\text{SliceCtbAddrRS} = (\text{slice_address} \gg \text{SliceGranularity})$$

【0133】

Zスキャン順序において最小符号化ブロック粒度におけるスライスにおいて第1の符号化ブロックのアドレスを指定する、変数SliceCbAddrZSは、以下のように導出される。

40

$$\text{SliceCbAddrZS} = \text{slice_address} \ll ((\log 2_diff_max_min_coding_block_size - \text{SliceGranularity}) \ll 1)$$

【0134】

スライス復号化は、可能な最大の符号化ユニットで、または、他の用語において、スライス開始座標で、CTUで、開始する。

【0135】

first__slice__in__pic__flagは、スライスが画像の第1のスライ

50

スであるかどうかを示す。`first_slice_in_pic_flag`が1に等しい場合、変数`SliceCbAddrZS`および`SliceCtbAddrRS`は、両方とも0にセットされ、さらに、復号化は、画像において第1の符号化ツリー・ブロックで開始する。

【0136】

`pic_parameter_set_id`は、使用において画像パラメータ・セットを指定する。`pic_parameter_set_id`の値は、0から255の範囲にある。

【0137】

`num_entry_point_offsets`は、スライス・ヘッダにおいて`entry_point_offset[i]` 構文要素の数を指定する。`tiles_or_entropy_coding_sync_idc`が1に等しいときに、`num_entry_point_offsets`の値は、0から $(\text{num_tile_columns} - \text{num_tile_rows}) - 1$ の範囲にある。`tiles_or_entropy_coding_sync_idc`が2に等しいときに、`num_entry_point_offsets`の値は、0から`PicHeightInCtbs - 1`の範囲にある。存在しないときに、`num_entry_point_offsets`の値は、0に等しいと推測される。

【0138】

`offset_len_minus1 + 1`は、ビットにおいて、`entry_point_offset[i]` 構文要素の長さを指定する。

【0139】

`entry_point_offset[i]` は、バイトにおいて、*i* 番目のエントリ・ポイント・オフセットを指定し、さらに、`offset_len_minus1 + 1`のビットによって表される。スライス・ヘッダの後の符号化されたスライス・データは、0から`num_entry_point_offsets`の範囲のサブセット・インデックス値を有する、`num_entry_point_offsets + 1`のサブセットからなる。サブセット0は、符号化されたスライス・データの0から`entry_point_offset[0] - 1`のバイトからなり、1から`num_entry_point_offsets - 1`の範囲における`k`を有するサブセット`k`は、符号化されたスライス・データの`entry_point_offset[k - 1]` から`entry_point_offset[k] + entry_point_offset[k - 1] - 1`のバイトからなり、さらに、 $(\text{num_entry_point_offsets} - \text{num_entry_point_offsets_used})$ に等しいサブセット・インデックスを有する最後のサブセットは、符号化されたスライス・データの残りのバイトからなる。

【0140】

`tiles_or_entropy_coding_sync_idc`が1に等しく、さらに、`num_entry_point_offsets`が0よりも大きいときに、それぞれのサブセットは、正確に1つのタイルのすべての符号化されたビットを含み、さらに、サブセットの数（すなわち、`num_entry_point_offsets + 1`の値）は、スライスにおいてタイルの数以下である。

【0141】

注意— `tiles_or_entropy_coding_sync_idc`が1に等しいときに、それぞれのスライスは、1つのタイルのサブセット（エントリ・ポイントの信号伝達が不必要である場合）か、整数の完全なタイルを含まなければならない。

【0142】

`tiles_or_entropy_coding_sync_idc`が2に等しく、さらに、`num_entry_point_offsets`が0よりも大きいときに、0から`num_entry_point_offsets - 1`の範囲における`k`を有するそれぞれのサブセット`k`は、符号化ツリー・ブロックの正確に1つの行のすべての符号化さ

れたビットを含み、(num_entry_point_offsetsに等しいサブセット・インデックスを有する)最後のサブセットは、スライスに含まれる残りの符号化ブロックのすべての符号化されたビットを含み、残りの符号化ブロックは、符号化ツリー・ブロックの正確に1つの行か、符号化ツリー・ブロックの1つの行のサブセットからなり、さらに、サブセットの数(すなわち、num_entry_point_offsets+1の値)は、スライスにおいて符号化ツリー・ブロックの行の数に等しく、スライスにおいて符号化ツリー・ブロックの1つの行のサブセットもカウントされる。

【0143】

注意 tiles_or_entropy_coding_sync_idcが2に等しいときに、スライスは、符号化ツリー・ブロックの多数の行および符号化ツリー・ブロックの行のサブセットを含んでもよい。例えば、スライスが符号化ツリー・ブロック2.5行を含む場合、サブセットの数(すなわち、num_entry_point_offsets+1の値)は、3に等しい。

10

【0144】

対応する画像パラメータ・セットRBSP構文を、図22に示すように選択することができる。

【0145】

画像パラメータ・セットRBSPセマンティックは、以下のようになり得る：

【0146】

1に等しいdependent_slice_enabled_flagは、画像パラメータ・セットを参照する符号化された画像のためのスライス・ヘッダにおいて構文要素dependent_slice_flagの存在を指定する。0に等しいdependent_slice_enabled_flagは、画像パラメータ・セットを参照する符号化された画像のためのスライス・ヘッダにおいて構文要素dependent_slice_flagの不存在を指定する。tiles_or_entropy_coding_sync_idcが3に等しいときに、dependent_slice_enabled_flagの値は、1に等しい。

20

【0147】

0に等しいtiles_or_entropy_coding_sync_idcは、画像パラメータ・セットを参照するそれぞれの画像において1つのタイルだけがあることを指定し、画像パラメータ・セットを参照するそれぞれの画像において符号化ツリー・ブロックの行の第1の符号化ツリー・ブロックを復号化する前に呼び出されるコンテキスト変数のための特定の同期化プロセスがなく、さらに、画像パラメータ・セットを参照する符号化された画像のためのcabac_independent_flagおよびdependent_slice_flagの値は、両方とも1に等しくない。

30

【0148】

cabac_independent_flagおよびdependent_slice_flagが両方ともスライスのために1に等しいときに、スライスは、エントロピー・スライスであることに注意する。

【0149】

1に等しいtiles_or_entropy_coding_sync_idcは、画像パラメータ・セットを参照するそれぞれの画像において1つよりも多いタイルがあってもよいことを指定し、画像パラメータ・セットを参照するそれぞれの画像において符号化ツリー・ブロックの行の第1の符号化ツリー・ブロックを復号化する前に呼び出されるコンテキスト変数のための特定の同期化プロセスがなく、さらに、画像パラメータ・セットを参照する符号化された画像のためのcabac_independent_flagおよびdependent_slice_flagの値は、両方とも1に等しくない。

40

【0150】

2に等しいtiles_or_entropy_coding_sync_idcは、画像パラメータ・セットを参照するそれぞれの画像において1つのタイルだけがあること

50

を指定し、コンテキスト変数のための特定の同期化プロセスは、画像パラメータ・セットを参照するそれぞれの画像において符号化ツリー・ブロックの行の第1の符号化ツリー・ブロックを復号化する前に呼び出され、さらに、コンテキスト変数のための特定の暗記プロセスは、画像パラメータ・セットを参照するそれぞれの画像において符号化ツリー・ブロックの行の2つの符号化ツリー・ブロックを復号化した後に呼び出され、さらに、画像パラメータ・セットを参照する符号化された画像のための `cabac__independent__flag` および `dependent__slice__flag` の値は、両方とも1に等しくない。

【0151】

3に等しい `tiles__or__entropy__coding__sync__idc` は、画像パラメータ・セットを参照するそれぞれの画像において1つのタイルだけがあることを指定し、画像パラメータ・セットを参照するそれぞれの画像において符号化ツリー・ブロックの行の第1の符号化ツリー・ブロックを復号化する前に呼び出されるコンテキスト変数のための特定の同期化プロセスがなく、さらに、画像パラメータ・セットを参照する符号化された画像のための `cabac__independent__flag` および `dependent__slice__flag` の値は、両方とも1に等しくてもよい。

【0152】

`dependent__slice__enabled__flag` が0に等しいときに、`tiles__or__entropy__coding__sync__idc` は、3に等しくない。

【0153】

`tiles__or__entropy__coding__sync__idc` の値が符号化されたビデオ・シーケンス内で起動されるすべての画像パラメータ・セットのためのものであることは、ビットストリーム適合性の要件である。

【0154】

画像パラメータ・セットを参照するスライスごとに、`tiles__or__entropy__coding__sync__idc` が2に等しく、さらに、スライスにおいて第1の符号化ブロックが符号化ツリー・ブロックの行の第1の符号化ツリー・ブロックにおいて第1の符号化ブロックでないときに、スライスにおいて最後の符号化ブロックは、スライス・スライスにおいて第1の符号化ブロックと同じ符号化ツリー・ブロックの行に属する。

【0155】

`num__tile__columns__minus1+1` は、画像をパーティション分割するタイル列の数を指定する。

`num__tile__rows__minus1+1` は、画像をパーティション分割するタイル行の数を指定する。

【0156】

`num__tile__columns__minus1` が0に等しいときに、`num__tile__rows__minus1` は、0に等しくない。1に等しい `uniform__spacing__flag` は、列境界および同様に行境界が画像にわたって一様に分布されることを指定する。0に等しい `uniform__spacing__flag` は、列境界および同様に行境界が画像にわたって一様に分布されないが、構文要素 `column__width[i]` および `row__height[i]` を用いて明確に信号伝送されることを指定する。

【0157】

`column__width[i]` は、符号化ツリー・ブロックを単位にして *i* 番目のタイル列の幅を指定する。

【0158】

`row__height[i]` は、符号化ツリー・ブロックを単位にして *i* 番目のタイル行の高さを指定する。

【0159】

ベクトル `colWidth[i]` は、0 から `num_tile_columns_minus1` の範囲の列 `i` を有する CTB を単位にして `i` 番目のタイル列の幅を指定する。

【0160】

ベクトル `CtbAddrRStoTS[ctbAddrRS]` は、0 から $(picHeightInCtbs * picWidthInCtbs) - 1$ の範囲のインデックス `ctbAddrRS` を有する、ラスタ・スキャン順序における CTB アドレスからタイル・スキャン順序における CTB アドレスへの会話を指定する。

【0161】

ベクトル `CtbAddrTStoRS[ctbAddrTS]` は、0 から $(picHeightInCtbs * picWidthInCtbs) - 1$ の範囲のインデックス `ctbAddrTS` を有する、タイル・スキャン順序における CTB アドレスからラスタ・スキャン順序における CTB アドレスへの会話を指定する。

【0162】

ベクトル `TileId[ctbAddrTS]` は、0 から $(picHeightInCtbs * picWidthInCtbs) - 1$ の範囲の `ctbAddrTS` を有する、タイル・スキャン順序における CTB アドレスからタイル `id` への会話を指定する。

【0163】

`colWidth`、`CtbAddrRStoTS`、`CtbAddrTStoRS` および `TileId` の値は、CTB ラスタと入力として `PicHeightInCtbs` および `PicWidthInCtbs` を有するタイル・スキャンング会話プロセスとを呼び出すことによって導出され、さらに、出力は、`colWidth`、`CtbAddrRStoTS` および `TileId` に割り当てられる。

【0164】

ルマ (`luma`) ・サンプルを単位にして `i` 番目のタイル列の幅を指定する、`ColumnWidthInLumaSamples[i]` の値は、`lcolWidth[i]` に等しくセットされ、 $lcolWidth[i] < \log_2 CtbSize$ である。

【0165】

0 から `picWidthInMinCtbs - 1` の範囲の `x` および 0 から `picHeightInMinCtbs - 1` の範囲の `y` を有する、最小 CB を単位にして場所 (`x`、`y`) から Z スキャン順序において最小 CB アドレスへの会話を指定する、アレイ `MinCbAddrZS[x][y]` は、入力として `Log2MinCbSize`、`Log2CtbSize`、`PicHeightInCtbs`、`PicWidthInCtbs` およびベクトル `CtbAddrRStoTS` を有する Z スキャンング・順序アレイ初期化プロセスを呼び出すことによって導出され、さらに、出力は、`MinCbAddrZS` に割り当てられる。

【0166】

1 に等しい `loop_filter_across_tiles_enabled_flag` は、ループにおいてフィルタリング動作がタイル境界を越えて実行されることを指定する。0 に等しい `loop_filter_across_tiles_enabled_flag` は、ループにおいてフィルタリング動作がタイル境界を越えて実行されないことを指定する。ループにおいてフィルタリング動作は、デブロッキング・フィルタ、サンプル適応オフセットおよび適応ループ・フィルタ動作を含む。存在しないときに、`loop_filter_across_tiles_enabled_flag` の値は、1 に等しいと推測される。

【0167】

1 に等しい `cabac_independent_flag` は、スライスにおいて符号化ブロックの CABAC 復号化が前に復号化されたスライスのいかなる状態からも独立していることを指定する。0 に等しい `cabac_independent_flag` は、スライスにおいて符号化ブロックの CABAC 復号化が前に復号化されたスライスの状態から依存していることを指定する。存在しないときに、`cabac_independe`

10

20

30

40

50

`nt_flag`の値は、0に等しいと推測される。

【0168】

最小符号化ブロック・アドレスを有する符号化ブロックの利用可能性のための導出プロセスは、以下のようになり得る：

【0169】

このプロセスへの入力は、

ー `z` スキャン順序において最小符号化ブロック・アドレス `minCbAddrZS`

ー `z` スキャン順序において現在の最小符号化ブロック・アドレス `currMinCbAddrZS`

である。

10

【0170】

このプロセスの出力は、`z` スキャン順序 `cbAvailable` において最小符号化ブロック・アドレス `cbAddrZS` を有する符号化ブロックの利用可能性である。

【0171】

注意1ー 利用可能性の意味は、このプロセスが呼び出されるときに決定される。

注意2ー そのサイズに関係なく、いかなる符号化ブロックは、最小符号化ブロック・アドレスと関連付けられ、それは、`z` スキャン順序において最小符号化ブロック・サイズを有する符号化ブロックのアドレスである。

ー 以下の状況の1つ以上が真である場合、`cbAvailable` は、偽にセットされる。

20

ー `minCbAddrZS` は、0未満である

ー `minCbAddrZS` は、`currMinCbAddrZS` より大きい

ー 最小符号化ブロック・アドレス `minCbAddrZS` を有する符号化ブロックは、現在の最小符号化ブロック・アドレス `currMinCbAddrZS` を有する符号化ブロックとは異なるスライスに属し、さらに、現在の最小符号化ブロック・アドレス `currMinCbAddrZS` を有する符号化ブロックを含むスライスの `dependent_slice_flag` は、0に等しい。

ー 最小符号化ブロック・アドレス `minCbAddrZS` を有する符号化ブロックは、現在の最小符号化ブロック・アドレス `currMinCbAddrZS` を有する符号化ブロックとは異なるタイルに含まれる。

30

ー そうでなければ、`cbAvailable` は、真にセットされる。

【0172】

スライス・データのためのCABAC解析プロセスは、以下のようになり得る：

【0173】

このプロセスは、記述子 `ae(v)` を有する特定の構文要素を解析するときに呼び出される。

【0174】

このプロセスへの入力は、構文要素の値および従来の解析された構文要素の値の要請である。

【0175】

このプロセスの出力は、構文要素の値である。

40

【0176】

スライスのスライス・データの解析を開始するときに、CABAC解析プロセスの初期化プロセスが呼び出される。

【0177】

図23は、どのように空間隣接Tが現在の符号化ツリー・ブロック（情報）と関連する符号化ツリー・ブロック利用可能性導出プロセスを呼び出すために用いられるかについて示す。

【0178】

空間隣接ブロックT（図23）を含む符号化ツリー・ブロックの最小符号化ブロック・

50

アドレス、`ctbMinCbAddrT`は、以下のように現在の符号化ツリー・ブロックの左上のルマ（`luma`）・サンプルの位置（`x0`、`y0`）を用いて導出される。

$$x = x0 + 2 \ll \text{Log2CtbSize} - 1$$

$$y = y0 - 1$$

$$\text{ctbMinCbAddrT} = \text{MinCbAddrZS} [x \gg \text{Log2MinCbSize}] [y \gg \text{Log2MinCbSize}]$$

【0179】

変数`availableFlagT`は、入力として`ctbMinCbAddrT`を有する符号化ブロック利用可能性導出プロセスを呼び出すことによって得られる。

【0180】

指定されるように符号化ツリーの解析を開始するときに、以下の順序付けられたステップが適用される。

【0181】

算術復号化エンジンは、以下のように初期化される。

【0182】

`CtbAddrRS`が`slice_address`に等しい場合、`dependent_slice_flag`は1に等しく、さらに、`entropy_coding_reset_flag`は、0に等しく、以下が適用される。

CABAC解析プロセスの同期化プロセスは、入力として`TableStateIdxDS`および`TableMPSValDS`で呼び出される。

終了の前のバイナリ決定のための復号化プロセスが呼び出され、その後に算術復号化エンジンのための初期化プロセスが続く。

【0183】

そうでなければ、`tiles_or_entropy_coding_sync_idc`が2に等しく、さらに、`CtbAddrRS % PicWidthInCtbs`が0に等しい場合、以下が適用される。

`availableFlagT`が1に等しいときに、CABAC解析プロセスの同期化プロセスは、入力として`TableStateIdxWPP`および`TableMPSValWPP`で呼び出される。

終了の前のバイナリ決定のための復号化プロセスが呼び出され、その後に算術復号化エンジンのためのプロセスが続く。

【0184】

`cabac_independent_flag`が0に等しく、さらに、`dependent_slice_flag`が1に等しいときに、または、`tiles_or_entropy_coding_sync_idc`が2に等しいときに、暗記プロセスは、以下のように適用される。

`tiles_or_entropy_coding_sync_idc`が2に等しく、さらに、`CtbAddrRS % PicWidthInCtbs`が2に等しいときに、CABAC解析プロセスの暗記プロセスは、出力として`TableStateIdxWPP`および`TableMPSValWPP`で呼び出される。

`cabac_independent_flag`が0に等しく、`dependent_slice_flag`が1に等しく、さらに、`end_of_slice_flag`が1に等しいときに、CABAC解析プロセスの暗記プロセスは、出力として`TableStateIdxDS`および`TableMPSValDS`で呼び出される。

【0185】

構文要素の解析は、以下のように進む：

【0186】

構文要素の要請された値ごとに、2値化が導出される。

【0187】

構文要素および解析されたビン（`bin`）のシーケンスのための2値化が、復号化プロ

10

20

30

40

50

セス・フローを決定する。

【0188】

変数 `binIdx` によってインデックスが付けられる、構文要素の2値化のビン (`bin`) ごとに、コンテキスト・インデックス `ctxIdx` が導出される。

【0189】

`ctxIdx` ごとに、算術復号化プロセスが呼び出される。

【0190】

解析されたビン (`bin`) の結果として生じるシーケンス (`b0...bbinIdx`) は、それぞれのビン (`bin`) の復号化の後に2値化プロセスによって与えられるビン (`bin`) ・ストリングのセットと比較される。シーケンスが与えられたセットにおいてビン (`bin`) ・ストリングにマッチするときに、対応する値は、構文要素に割り当てられる。

10

【0191】

構文要素の値のための要請が構文要素 `pcm_flag` のために処理され、さらに、`pcm_flag` の復号化された値が1に等しい場合、復号化エンジンは、いかなる `pcm_alignment_zero_bit`、`num_subsequent_pcm`、すべての `pcm_sample_luma` および `pcm_sample_chroma` データの復号化の後に初期化される。

【0192】

このように、上述の説明は、図24に示すようなデコーダを明らかにする。参照符号5によって一般に示されるこのデコーダは、画像10がパーティション分割されるスライス14を単位にして画像10が符号化されるデータ・ストリーム12から画像10を再構成し、デコーダ5は、スライス順序16に従ってデータ・ストリーム12からスライス14を復号化するように構成される。当然、デコーダ5は、スライス14を連続的に復号化するために制限されない。むしろ、デコーダ5は、スライス14にパーティション分割する画像10が波面並列処理に適切であるという条件で、スライス14を復号化するために波面並列処理を用いることができる。したがって、デコーダ5は、例えば、上述されさらに以下にも記載されているように、波面処理を可能にするためにスライス順序16を考慮することによってスライス14の復号化を開始して、スライス14を千鳥状に並列に復号化することができる、デコーダであってもよい。

20

30

【0193】

デコーダ5は、少なくとも2つのモード20および22の1つに従って現在のスライスを復号化するために、スライス14の現在のスライス内で構文要素部分18に応答する。少なくとも2つのモードの第1のモード、すなわちモード20に従って、現在のスライスは、スライス境界を越える、すなわち図24において点線を越える、コンテキストの導出を含むコンテキスト適応エントロピー復号化を用いて、すなわち他の「スライス順序16において前のスライス」の符号化／復号化から生じる情報を用いることによって、データ・ストリーム12から復号化される。さらに、第1のモード20を用いてデータ・ストリーム12から現在のスライスの復号化は、コーデックのシンボル確率の連続的な更新、および前に復号化されたスライスのシンボル確率の保存された状態に依存する現在のスライスの復号化の開始時のシンボル確率の初期化を含む。そのような依存性は、例えば、「コーデック変数のための同期化プロセス」と関連して、上述されている。最後に、第1のモード20は、スライス境界を越える予測復号化も含む。スライス境界を越えるそのような予測復号化は、例えば、スライス境界、すなわち「スライス順序16において」前のスライスのすでに再構成されたサンプル値に基づく現在のスライス内の予測サンプル値を、越えるイントラ予測、または、例えば運動ベクトルの予測などのスライス境界を越える符号化パラメータの予測、予測モード、符号化モードなどを含むことができる。

40

【0194】

第2のモード22に従って、デコーダ5は、コンテキスト適応エントロピー復号化を用いるが、スライス境界を越えないようにコンテキストの導出を制限して、データ・ストリ

50

ーム12から、現在のスライス、すなわち現在復号化されるスライスを復号化する。例えば、現在のスライス内でブロックに関連する特定の構文要素のためのコンテキストを導出するために用いられる隣接した位置のテンプレートが、隣接したスライスに拡張し、それによって現在のスライスのスライス境界を越える場合、例えば隣接したスライスのこの隣接した部分の対応する構文要素の値などのように、隣接したスライスのそれぞれの部分の対応する属性は、現在のスライスおよび隣接したスライス間の相互依存性を抑制するためにデフォルト値にセットされる。コンテキストのシンボル確率の連続的な更新は、それが第1のモード20における場合であるように起こり得る一方、第2のモード22におけるシンボル確率の初期化は、いかなる前に復号化されたスライスからも独立している。さらに、予測復号化は、スライス境界を越えないように予測復号化を制限して実行される。

10

【0195】

図24の説明および以下の説明の理解を容易にするために、図24と比較してより構造的な感覚においてデコーダ5の可能な実施を示す図25を参照する。それが図24における場合であるように、デコーダ5は、例えば、予測残差および予測パラメータを得るためにデータ・ストリームを復号化するためのコンテキスト適応エントロピー復号化を用いる予測デコーダである。

【0196】

図25に示すように、デコーダ5は、エントロピー・デコーダ24、逆量子化および逆変換モジュール26、例えば、加算器として、図25に示すように、実施されるコンバイナ28、および予測器28を含むことができる。エントロピー・デコーダ24、モジュール26および加算器27は、それらの言及する順序においてデコーダ5の入力および出力間に直列的に接続され、さらに、予測器28は、コンバイナ27とともに予測ループを形成するために加算器28の出力およびそのさらなる入力間に接続される。そのため、デコーダ24は、予測器28の符号化パラメータ入力にさらに接続されるその出力を有する。

20

【0197】

図25は、デコーダが現在の画像を連続的に復号化するという印象を提供するにもかかわらず、デコーダ5は、例えば、画像10を並列に復号化するために実施され得る。デコーダ5は、例えば、それぞれが図25において要素24-28に従って作動する複数のコアを含むことができる。しかしながら、並列処理は、任意であり、さらに、連続的に作動するデコーダ5は、エントロピー・デコーダ24の入力に入ってくるデータ・ストリームを復号化することもできる。

30

【0198】

現在の画像10を連続的にまたは並列に復号化する今述べた能力を効率的に達成するために、デコーダ5は、画像10を復号化するために、符号化ブロック30を単位にして作動する。符号化ブロック30は、例えば、符号化ツリー・ブロックまたは最大の符号化ブロック32が例えばクワッドツリー・パーティション分割などの再帰的なマルチツリー・パーティション分割によってパーティション分割される、リーフ・ブロックである。次に、コード・ツリーブロック32は、これらのコード・ツリーブロック32への画像10のレギュラー・パーティション分割を形成するために、列および行に規則的に配置され得る。図25において、コード・ツリーブロック32は、実線で示されるが、符号化ブロック30は、点線で示される。例示の目的のために、単に1つのコード・ツリーブロック32は、符号化ブロック30にさらにパーティション分割されることを示し、その代わりに、他のコード・ツリーブロック32は、符号化ブロックを直接的に形成するためにさらにパーティション分割されないことを示す。データ・ストリーム12は、どのように画像10がコード・ブロック30にパーティション分割されるかについて信号伝達する構文部分を含むことができる。

40

【0199】

データ・ストリーム12は、符号化ブロック30ごとに、どのようにモジュール24-28がその符号化ブロック30内の画像コンテンツを回復するかについて明らかにする構文要素を伝達する。例えば、これらの構文要素は、以下を含む：

50

1) 任意に、符号化ブロック 30 を予測ブロックにさらにパーティション分割するパーティション分割データ、

2) 任意に、符号化ブロック 30 を残差および／または変換ブロックにさらにパーティション分割するパーティション分割データ、

3) 予測モードが符号化ブロック 30 のための予測信号を導出するために用いられるかについて信号伝達する予測モードであって、この予測モードが信号伝達される粒度は、符号化ブロック 30 および／または予測ブロックに依存することができる。

4) 予測パラメータは符号化ブロックごとに信号伝達され得り、または、存在する場合、送られるある種の予測パラメータを有する予測ブロックごとに、例えば、予測モードに依存する。可能な予測モードは、例えば、イントラ予測および／またはインター予測を含むことができる。

10

5) 他の構文要素は、例えば、予測信号および／または再生される再構成された信号を得るために符号化ブロック 30 で画像 10 をフィルタリングするためのフィルタリング情報などのように存在されてもよい。

6) 最後に、特に変換係数の形の残差情報は、符号化ブロック 30 のためのデータ・ストリームに含まれ得り、残差ブロックを単位にして、残差データは、信号伝達され得り、残差ブロックごとに、存在する場合、スペクトル分解は、例えば、上述した変換ブロックを単位にして実行され得る。

【0200】

エントロピー・デコーダ 24 は、データ・ストリームから今述べた構文要素を得る役割を果たす。この目的のために、エントロピー・デコーダ 24 は、コンテキスト適応エントロピー復号化を用いる。すなわち、エントロピー・デコーダ 24 は、いくつかのコンテキストを提供する。データ・ストリーム 12 から特定の構文要素を導出するために、エントロピー・デコーダ 24 は、可能なコンテキスト中の特定のコンテキストを選択する。可能なコンテキスト中の選択は、現在の構文要素が属する画像 10 の部分の近辺の属性に応じて実行される。可能なコンテキストのそれぞれのために、エントロピー・デコーダ 24 は、シンボル確率、すなわち、エントロピー・デコーダ 24 が作動するシンボル・アルファベットのそれぞれの可能なシンボルのための確率推定を管理する。「管理する」ことは、それぞれのコンテキストに関連付けられるシンボル確率を実際の画像コンテキストに適応するために、コンテキストのシンボル確率の上述した連続的な更新を含む。この対策によ

20

30

【0201】

近辺の属性が例えば現在の符号化ブロック 30 などの画像 10 の現在の部分の再構成に影響を与える別の環境は、予測器 28 内の予測復号化である。予測は、現在の符号化ブロック 30 内の予測コンテンツだけに制限されないが、例えば予測パラメータなどの現在の符号化ブロック 30 のためのデータ・ストリーム 12 内に含まれるパラメータ、パーティション分割データまたは変換係数の予測を含むこともできる。すなわち、予測器 28 は、データ・ストリーム 12 からモジュール 26 によって得られるような予測残差とその後結合される書き込まれた信号を得るために、上述した近辺から画像コンテンツまたはそのようなパラメータを予測することができる。パラメータを予測する場合、予測器 28 は、予測パラメータの実際の値を得るために予測残差としてデータ・ストリーム内に含まれる構文要素を用いることができる。予測器 28 は、コンバイナ 27 において予測残差と結合される今述べた予測信号を得るために、後の予測パラメータ値を用いる。

40

【0202】

上述した「近辺」は、現在エントロピー復号化される構文要素または現在予測される構文要素が属する現在の部分の環境の左上部分を主にカバーする。図 25 において、そのような近辺は、1 つの符号化ブロック 30 のために例示的に 34 で示される。

【0203】

符号化／復号化順序は、符号化ブロック 30 中で定義される：最も粗いレベルで、画像 10 のコード・ツリーブロック 32 は、ここでは上から下まで行ごとに導くラスト・スキ

50

ヤンとして示される、スキャン順序36においてスキャンされる。それぞれのコード・ツリーブロック内で、符号化ブロック30は、それぞれの階層レベルにおいて、コード・ツリーブロック32が上から下まで行ごとに導くラスタ・スキャンにおいても実質的にスキャンされるように、深さ優先トラバーサル順序においてスキャンされる。

【0204】

符号化ブロック30中で定義される符号化順序は、近辺34が主に符号化順序に従って復号化をすでに受けた画像10の部分をカバーするという点で、コンテキストを選択しおよび／または空間予測を実行するために、近辺において属性を導出するために用いられる近辺34の定義と調和する。近辺34の部分が画像10の利用できない部分をカバーするときはいつでも、デフォルト・データは、例えば、その代わりに用いられる。例えば、近辺テンプレート34は、画像10の外側に拡張することができる。しかしながら、別の可能性は、近辺34が隣接したスライスに拡張するという点である。

10

【0205】

スライス分割、例えば、符号化ブロック30に沿って定義される符号化／復号化順序に沿った画像10、すなわちそれぞれのスライスは、上述した符号化ブロック順序に沿った符号化ブロック30の連続的な中断しないシーケンスである。図25において、スライスは、1点鎖線14で示される。スライス14中で定義される順序は、上述のような連続的な符号化ブロック30の実行のそれらの構成から生じる。特定のスライス14の構文要素部分18はそれが第1のモードにおいて復号化されることを示す場合、エントロピー・デコーダ24は、コンテキスト適応エントロピー復号化がスライス境界を越えるコンテキストを導出することを可能にする。すなわち、空間近辺34は、現在のスライス14に関してエントロピー復号化データにおいてコンテキストを選択するために用いられる。図25の場合、例えば、スライス・ナンバー3は、現在復号化されたスライスであってもよく、さらに、符号化ブロック30に関してエントロピー復号化構文要素またはそこに含まれるいくつかの部分において、エントロピー・デコーダ24は、例えばスライス・ナンバー1などの隣接したスライス内で復号化部分から生じる属性を用いることができる。予測器28は、同じように作動する：第1のモード20にあるスライスのために、予測器28は、現在のスライスを囲むスライス境界を越える空間予測を用いる。

20

【0206】

しかしながら、それに関連付けられる第2のモード22を有する、すなわち構文要素部分18が第2のモード22を示す、スライスのために、エントロピー・デコーダ24および予測器28は、現在のスライス内だけに存在する部分に関連する属性に依存するために、エントロピー・コンテキストの導出および予測復号化を制限する。明らかに、符号化効率は、この制限を受ける。一方、第2のモード22のスライスは、スライスのシーケンス間の相互依存性を破壊することを可能にする。したがって、第2のモード22のスライスは、画像10内にまたは画像10が再同期化ポイントを可能にするために属するビデオ内に散在され得る。しかしながら、それぞれ画像10が第2のモード22において少なくとも1つのスライスを有することは必要でない。

30

【0207】

すでに上述したように、第1および第2のモード20および22も、シンボル確率のそれらの初期化において異なる。第2のモード22において符号化されるスライスは、いかなる前に復号化されたスライス、すなわちスライス中で定義される順序という意味において前に復号化されるスライス、から独立した確率を再初期化するエントロピー・デコーダ24をもたらず。シンボル確率は、例えば、エンコーダおよびデコーダ側にとって知られているデフォルト値にセットされ、または、初期化値は、第2のモード22において符号化されるスライス内に含まれる。

40

【0208】

すなわち、第2のモード22において符号化され／復号化されるスライスのために、シンボル確率の適応は、常にこれらのスライスの開始から直ちに開始する。したがって、適応精度は、これらのスライスの開始時にこれらのスライスのために悪い。

50

【0209】

事態は、第1のモード20において符号化され／復号化されるスライスにおいて異なる。後のスライスのために、エントロピー・デコーダ24によって実行されるシンボル確率の初期化は、前に復号化されたスライスのシンボル確率の保存された状態に依存する。第1のモード20において符号化され／復号化されるスライスが、例えば、画像10の左側以外に、すなわちラスタ・スキャン36が次の行の下区に進む前に行ごとに実行を開始する側ではない側に、配置される、その開始を有するときはいつでも、直前のスライスのエントロピー復号化の最後に生じるようなシンボル確率が適応される。これは、例えば、スライス・ナンバー4のための矢印38によって図2において示される。スライス・ナンバー4は、画像10の右側および左側の間に、したがって、シンボル確率を初期化する際に、どこかにその開始を有し、エントロピー・デコーダ24は、シンボル確率を初期化する際に、直前のスライス、すなわちスライス・ナンバー3のエントロピー復号化において得られたシンボル確率を、その最後まで、すなわちスライス3のエントロピー復号化の間にシンボル確率の連続的な更新を含むその最後まで適応する。

10

【0210】

それに関連付けられる第2のモード22を有するが、例えばスライス・ナンバー5などの画像10の左側で開始を有するスライスは、直前のスライス・ナンバー4のエントロピー復号化を終了した後に得られるようなシンボル確率を適応しなく、なぜなら、これは、デコーダ5が波面処理の使用によって画像10を並列的に復号化するのを妨ぐからである。むしろ、上述のように、エントロピー・デコーダ24は、矢印40で示すように符号化／復号化順序36において直前のコード・ツリーブロック行において、符号化／復号化順序36において第2のコード・ツリーブロック32のエントロピー復号化を終了した後に得られるようなシンボル確率を適応する。

20

【0211】

図25において、例えば、画像10は、3行のコード・ツリーブロックおよび4列の符号化ツリー・ルート・ブロック32に例示的にパーティション分割され、さらに、それぞれのコード・ツリーブロック行は、2つのスライス14に再分割され、その結果、すべての第2のスライスの開始は、それぞれのコード・ツリールート・ブロック行の符号化ユニット順序において第1の符号化ユニットと一致する。したがって、エントロピー・デコーダ24は、第1または最上のコード・ツリー・ルート・ブロック行から開始し、そして第2、それから第3に、これらのコード・ツリー・ルート・ブロック行を千鳥状に復号化することを開始して、それぞれのコード・ツリー・ルート・ブロック行を並列に復号化することによって、復号化画像10において波面処理を用いることができる。

30

【0212】

当然、さらなる符号化ブロック30への再帰的な方法においてブロック32のパーティション分割は、任意であり、したがって、より一般的な感覚において、ブロック32は、同様に「符号化ブロック」と呼ばれ得る。すなわち、より一般に言って、画像10は、行および列に配置されさらに互いに定義されるラスタ・スキャン順序36を有する符号化ブロック32にパーティション分割され得り、さらに、デコーダ5は、サブセットがスライス順序に従ってラスタ・スキャン順序36に沿って互いに続くように、それぞれのスライス14をラスタ・スキャン順序36において符号化ブロック32の連続的なサブセットと関連付けるために考慮され得る。

40

【0213】

また、上述の説明から明らかなように、デコーダ5またはより詳しくはエントロピー・デコーダ24は、ラスタ・スキャン順序36に従って符号化ブロック行において第2の符号化ブロックまでいかなるスライスのコンテキスト適応エントロピー復号化において得られるようなシンボル確率を保存するように構成され得る。それに関連付けられる第1のモード20を有する現在のスライスのコンテキスト適応エントロピー復号化のためのシンボル確率を初期化する際に、デコーダ5またはより詳しくはエントロピー・デコーダ24は、現在のスライスと関連付けられる符号化ブロック32の連続的なサブセットの第1の符

50

号化ブロック 3 2 がラスタ・スキャン順序 3 6 に従って符号化ブロック行において第 1 の符号化ブロック 3 2 であるかどうかについてチェックする。そうである場合、現在のスライスのコンテキスト適応エントロピー復号化のためのシンボル確率は、矢印 4 0 に関して説明されるように、すなわちラスタ・スキャン順序 3 6 に従って符号化ブロック行において第 2 の符号化ブロックまで前に復号化されたスライスのコンテキスト・エントロピー復号化において得られるような保存されたシンボル確率に応じて、初期化される。そうでない場合、現在のスライスのコンテキスト適応エントロピー復号化のためのシンボル確率の初期化は、前に復号化されたスライスの最後まで前に復号化されたスライスのコンテキスト適応エントロピー復号化において得られるようなシンボル確率に応じて、すなわち矢印 3 8 に従って、実行される。また、3 8 に従う初期化の場合、スライス順序 3 6 において直前のスライスのエントロピー復号化の最後に保存された状態が意味されるが、初期化 4 0 の場合、それは、前に復号化されたスライスがブロック順序 3 6 においてブロック 3 2 の直前の行の第 2 のブロックの最後を含むことである。

10

【0 2 1 4】

図 2 4 において点線で示すように、デコーダは、少なくとも 3 つのモードの 1 つに従って現在のスライスを復号化するためにスライス 1 4 の現在のスライス内で構文要素部分 1 8 に応答するように構成され得る。すなわち、第 3 のモード 4 2 が、他のもの 2 0 および 2 2 のそばにあってもよい。第 3 のモード 4 2 は、スライス境界を越える予測が可能にされるという点で第 2 のモード 2 2 と異なってもよいが、エントロピー符号化／復号化は、スライス境界を越えないようにまだ制限される。

20

【0 2 1 5】

上に、2 つの実施形態が、構文要素部分 1 8 に関して示されている。下のテーブルは、これらの 2 つの実施形態を要約する。

【0 2 1 6】

	実施形態 1	実施形態 2
構文要素部分	<code>dependent_slice_flag</code> , <code>no_cabac_reset_flag</code>	<code>dependent_slice_flag</code>
モード 1	<code>dependent_slice_flag = 1</code> , <code>no_cabac_reset_flag = 1</code>	<code>dependent_slice_flag = 1</code> ,
モード 2	<code>dependent_slice_flag = 0</code>	<code>dependent_slice_flag = 0</code>
モード 3	<code>dependent_slice_flag = 1</code> , <code>no_cabac_reset_flag = 0</code>	<code>dependent_slice_flag = 1</code> , <code>cabac_independent_flag = 1</code> , <code>tiles_or_entropy_coding_sync_idc = 3</code>

30

【0 2 1 7】

1 つの実施形態において、構文要素部分 1 8 は、個々に `dependent_slice_flag` によって形成され、その一方で、他の実施形態において、`dependent_slice_flag` および `no_cabac_reset_flag` の結合は、構文要素部分を形成する。前に復号化されたスライスのシンボル確率の保存された状態に応じたシンボル確率の初期化に関する限り、コンテキスト変数のための同期化プロセスを参照する。特に、デコーダは、`last_ctb_cabac_init_flag = 0` および `tiles_or_entropy_coding_sync_idc = 2` である場合、ラスタ・スキャン順序に従って行において第 2 の符号化ブロックまで前に復号化されたスライスのコンテキスト適応エントロピー復号化において得られるようなシンボル確率を保存し、さらに、第 1 のモードに従って現在のスライスのコンテキスト適応エントロピー復号化のためのシンボル確率を初期化する際に、現在のスライスと関連付けられる符号化ブロックの連続的なサブセットの第 1 の符号化ブロックがラスタ・スキャン順序に従って行において第 1 の符号化ブロックであるかどうかについてチェックし、さらに、そうである場合、ラスタ・スキャン順序に従って行において第 2 の符号化ブロックまで前に復号化されたスライスのコンテキスト適応エントロピー復号化において得られるような保存さ

40

50

【 0 2 1 8 】

50

適応エントロピー復号化と、スライス境界を越える予測復号化とを用いてデータストリームから現在のスライスを復号化するように構成され、第1および第3のモードの1つは、構文要素、すなわち `cabac_independent_flag` に応じて選択される。デコーダは、さらに、すなわち `tiles_or_entropy_coding_sync_idc = 0, 1, および 3 (cabac_independent_flag = 0 であるときに「3」)` である場合、前に復号化されたスライスの最後まで前に復号化されたスライスのコンテキスト適応エントロピー復号化において得られるようなシンボル確率を保存し、さらに、第1のモードに従って現在のスライスのコンテキスト適応エントロピー復号化のためのシンボル確率を初期化する際に、保存されるシンボル確率に応じて現在のスライスのコンテキスト適応エントロピー復号化のためのシンボル確率を初期化するように構成され得る。デコーダは、すなわち `tiles_or_entropy_coding_sync_idc = 1` である場合、第1および第2のモードにおいて、画像が再分割されるタイル内で予測復号化を制限するように構成され得る。

【0219】

当然、エンコーダは、したがって、デコーダが上述の利点を得ることを可能にするために、上述の構文をセットすることができる。エンコーダは、例えばマルチコア、エンコーダなどのような並列処理であってもよいが、必要ではない。スライス14を単位にして画像10をデータストリーム12に符号化するために、エンコーダは、スライス順序16に従ってスライス14をデータストリーム12に符号化するように構成される。エンコーダは、構文要素部分が少なくとも2つのモード20、22の1つに従って符号化される現在のスライスを信号伝達するようにスライスの現在のスライスのために、構文要素部分18を決定し、さらに、それに構文要素部分18を符号化し、さらに、現在のスライスが少なくとも2つのモードの第1のモード20に従って符号化される場合、スライス境界を越えるコンテキストの導出、コンテキストシンボル確率の連続的な更新および前に符号化されたスライスのシンボル確率の保存された状態に応じたシンボル確率の初期化38、40を含むコンテキスト適応エントロピー符号化24と、スライス境界を越える予測符号化とを用いて現在のスライスをデータストリーム12に符号化し、さらに、現在のスライスが少なくとも2つのモードの第2のモード22に従って符号化される場合、スライス境界を越えないようにコンテキストの導出を制限し、コンテキストのシンボル確率の連続的な更新およびいかなる前に符号化されたスライスから独立したシンボル確率の初期化を有するコンテキスト適応エントロピー符号化と、スライス境界を越えないように予測符号化を制限する予測符号化とを用いて現在のスライスをデータストリーム12に符号化する。画像10が行および列に配置されさらに互いに定義されるラスタ・スキャン順序36を有する符号化ブロック32においてパーティション分割され得ると同時に、エンコーダは、サブセットがスライス順序に従ってラスタ・スキャン順序36に沿って互いに続くように、それぞれのスライス14をラスタ・スキャン順序36において符号化ブロック32の連続的なサブセットと関連付けるように構成され得る。エンコーダは、ラスタ・スキャン順序36に従って行において第2の符号化ブロック32まで前に符号化されたスライスのコンテキスト適応エントロピー符号化において得られるようなシンボル確率を保存し、さらに、第1のモードに従って現在のスライスのコンテキスト適応エントロピー符号化のためのシンボル確率を初期化する際に、現在のスライスと関連付けられる符号化ブロック32の連続的なサブセットの第1の符号化ブロックがラスタ・スキャン順序に従って行において第1の符号化ブロック32であるかどうかについてチェックし、さらに、そうである場合、ラスタ・スキャン順序36に従って行において第2の符号化ブロックまで前に符号化されたスライスのコンテキスト適応エントロピー符号化において得られるような保存されたシンボル確率に応じて現在のスライスのコンテキスト適応エントロピー符号化のためのシンボル確率を初期化40し、さらに、そうでない場合、前に符号化されたスライスの最後まで前に復号化されたスライスのコンテキスト適応エントロピー符号化において得られるようなシンボル確率に応じて現在のスライスのコンテキスト適応エントロピー符号化のためのシンボル確率を初期化38するように構成され得る。エンコーダは、現在のスライスが少

10

20

30

40

50

なくとも3つのモードの1つに従って、すなわち第1のモード(20)および第3のモード(42)の1つまたは第2のモード(22)において、それに符号化されるように信号伝達されるように、構文要素部分(18)をスライス(14)の現在のスライスに符号化するように構成され得り、エンコーダは、**第3のモード(42)に従って、スライス境界を越えないようにコンテキストの導出を制限し、コンテキストのシンボル確率の連続的な更新およびいかなる前に符号化されたスライスから独立したシンボル確率の初期化を有するコンテキスト適応エントロピー符号化と、スライス境界を越える予測符号化とを用いて、現在のスライスをデータストリームに符号化するように構成され、エンコーダは、例えば、構文要素、すなわち `c a b a c _ i n d e p e n d e n t _ f l a g` を用いて第1および第3のモードの1つであるかを区別する。エンコーダは、例えば `d e p e n d e n t _ s l i c e s _ p r e s e n t _ f l a g` などの一般的な構文要素を決定し、さらに、一般的な構文要素に応じて少なくとも2つの一般的なオペレーティング・モードの1つにおいて作動して、すなわち、第1の一般的なオペレーティング・モードに従って、スライスごとに構文要素部分の符号化を実行し、さらに、第2の一般的なオペレーティング・モードに従って、第1のモード以外の少なくとも2つのモードの異なる1つを必然的に用いて、一般的な構文要素をデータストリームに書き込むように構成され得る。エンコーダは、第1および第2のモードに従って、現在のスライスの開始から最後までシンボル確率の連続的な更新を必然的にかつ途切れずに続けるように構成され得る。エンコーダ `a y` は、前に符号化されたスライスの最後まで前に符号化されたスライスのコンテキスト適応エントロピー符号化において得られるようなシンボルに確率を保存し、さらに、第1のモードに従って現在のスライスのコンテキスト適応エントロピー符号化のためのシンボル確率を初期化する際に、保存されるシンボル確率に応じて現在のスライスのコンテキスト適応エントロピー符号化のためのシンボル確率を初期化するように構成される。さらに、エンコーダは、第1および第2のモードにおいて、画像が再分割されるタイル内で予測符号化を制限することができる。

【0220】

エンコーダの可能な構造は、完全性のために図26において表される。予測器70は、ほとんど予測器28と同様に作動し、すなわち予測を実行するが、最適化によって、例えば、予測パラメータおよびモードを含む符号化パラメータも決定する。モジュール26および27も、デコーダにおいて生じる。減算器72は、無損失予測残差を決定し、それは、量子化の使用によって、さらに、任意に、スペクトル分解変換を用いて、変換および量子化モジュール74においてロッシーに符号化される。エントロピー・コーダ76は、コンテキスト適応エントロピー符号化を実行する。

【0221】

上述の具体的な構文例に加えて、異なる例が、以下に用いられる用語と上述に用いられる用語間のコンコーダンスを示して以下に概説される。

【0222】

特に、上述に概説しないで、依存スライスは、それがその境界の外側から知られる知識を利用することを可能にするという点で「依存」というだけでなく、例えば上述に概説されるように、より急速に適応されるエントロピー・コンテキストを有し、または、その境界を越える許容のためより良好な空間予測を達成する。むしろ、画像をスライスに分割することによってスライス・ヘッダを定義するために費やされるべきレート・コストを節約するために、依存スライスは、前のスライスからスライス・ヘッダ構文の部分を適応し、すなわち、このスライス構文ヘッダ部分は、依存スライスのために再び送信されない。これは、例えば、図16において100でさらに図21において102で示され、それに従って、スライス・タイプは、例えば、前のスライスから適応される。この対策によって、例えば独立スライスおよび依存スライスなどのスライスへの画像の再分割は、高価なビット消費に関してより安価である。

【0223】

それは、以下の概説された例において、わずかに異なる言い回しをもたらす今述べた依

10

20

30

40

50

存性である：スライスは、スライス・ヘッダ構文が個々にセット可能である画像のユニット部分として定義される。したがって、スライスは、上述の命名法を用いて、現在、独立スライス・セグメントと呼ばれる、1つの独立／レギュラー／ノーマル・スライス、および、上述の命名法を用いて、現在、依存スライス・セグメントと呼ばれる、0、1または2以上の依存スライスからなる。

【0224】

図27は、例えば、2つのスライスにパーティション分割される画像を示し、1つは、スライス・セグメント14₁～14₃で形成され、さらに、他の1つは、単にスライス・セグメント14₄だけで形成される。インデックス1～4は、符号化順序においてスライス順序を示す。図28aおよび図28bは、2つのタイルへの画像10の再分割の場合において異なる例を示し、図28aの場合、1つのスライスは、インデックスが符号化順序において上がる、両方のタイル50₁および50₂をカバーして、全5つのスライス・セグメント14で形成され、さらに、図28bの場合、2つのスライスは、それぞれタイル50₁を再分割して、スライス・セグメント14₁および14₂および14₃および14₄で形成され、別のスライスは、タイル50₂をカバーして、スライス・セグメント14₅～14₆で形成される。

10

【0225】

定義は、以下のようになり得る：

【0226】

依存スライス・セグメント：スライス・セグメント・ヘッダのいくつかの構文要素の値が復号化順序において前の独立スライス・セグメントのための値から推測されるスライス・セグメントであって、上述の実施形態において、依存スライスと前に呼ばれる。

20

【0227】

独立スライス・セグメント：スライス・セグメント・ヘッダの構文要素の値が前のスライス・セグメントのための値から推測されないスライス・セグメントであって、上述の実施形態において、前にノーマル・スライスと前に呼ばれる。

【0228】

スライス：1つの独立スライス・セグメントおよび同じアクセス・ユニット／画像内で次の独立スライス・セグメント（もしあれば）に先行するすべての後の依存スライス・セグメント（もしあれば）に含まれる符号化ツリー・ユニットの整数。

30

【0229】

スライス・ヘッダ：現在のスライス・セグメントでありまたは現在の依存スライス・セグメントに先行する独立スライス・セグメントである独立スライス・セグメントのスライス・セグメント・ヘッダ。

【0230】

スライス・セグメント：タイル・スキャンにおいて連続的に順序付けられさらに単一のNALユニットに含まれる符号化ツリー・ユニットの整数であって、スライス・セグメントへのそれぞれの画像の再分割がパーティション分割である。

【0231】

スライス・セグメント・ヘッダ：スライス・セグメントにおいて表される第1のまたはすべての符号化ツリー・ユニットに関連するデータ要素を含む符号化されたスライス・セグメントの部分。

40

【0232】

「モード」20および22、すなわち「依存スライス・セグメント」および「独立スライス・セグメント」の信号伝達は、以下のようになり得る：

【0233】

例えばPPSなどのいくつかの追加のNALユニットにおいて、構文要素は、依存スライスの使用が特定の画像のためのシーケンスの特定の画像のために形成されるかどうかについて信号伝送するために用いられ得る：

【0234】

50

1に等しい`dependent_slice_segments_enabled_flag`は、スライス・セグメント・ヘッダにおいて構文要素`dependent_slice_segment_flag`の存在を指定する。0に等しい`dependent_slice_segments_enabled_flag`は、スライス・セグメント・ヘッダにおいて構文要素`dependent_slice_segment_flag`の不存在を指定する。

【0235】

`dependent_slice_segments_enabled_flag`は、前に記載された`dependent_slices_present_flag`の範囲と同様である。

10

【0236】

同様に、`dependent_slice_flag`は、スライスに関して異なる命名法を占めるために、`dependent_slice_segment_flag`と呼ばれ得る。

【0237】

1に等しい`dependent_slice_segment_flag`は、現在のスライス・セグメントのヘッダに存在しないそれぞれのスライス・セグメント・ヘッダ構文要素の値がスライス・ヘッダ、すなわち前の独立スライス・セグメントのスライス・セグメント・ヘッダにおいて対応するスライス・セグメント・ヘッダ構文要素の値に等しいと推測されることを指定する。

20

【0238】

例えば画像レベルなどの同じレベルにおいて、以下の構文要素が含まれ得る：

【0239】

1に等しい`entropy_coding_sync_enabled_flag`は、コンテキスト変数のための特定の同期化プロセスがPPSを参照するそれぞれの画像においてそれぞれのタイルにおける符号化ツリー・ブロックの行の第1の符号化ツリー・ブロックを含む符号化ツリー・ユニットを復号化する前に呼び出されることを指定し、さらに、コンテキスト変数のための特定の記憶プロセスがPPSを参照するそれぞれの画像においてそれぞれのタイルにおける符号化ツリー・ブロックの行の第2の符号化ツリー・ブロックを含む符号化ツリー・ユニットを復号化した後に呼び出される。0に等しい`entropy_coding_sync_enabled_flag`は、コンテキスト変数のための特定の同期化プロセスがPPSを参照するそれぞれの画像においてそれぞれのタイルにおける符号化ツリー・ブロックの行の第1の符号化ツリー・ブロックを含む符号化ツリー・ユニットを復号化する前に呼び出されることが必要とされないことを指定し、さらに、コンテキスト変数のための特定の記憶プロセスがPPSを参照するそれぞれの画像においてそれぞれのタイルにおける符号化ツリー・ブロックの行の第2の符号化ツリー・ブロックを含む符号化ツリー・ユニットを復号化した後に呼び出されることが必要とされない。

30

`entropy_coding_sync_enabled_flag`の値がCVS内で起動されるすべてのPPSのためのものであることは、ビットストリーム適合性の要件である。

40

`entropy_coding_sync_enabled_flag`が1に等しく、さらに、スライスにおいて第1の符号化ツリー・ブロックがタイルにおいて符号化ツリー・ブロックの行の第1の符号化ツリー・ブロックでないときに、スライスにおいて最後の符号化ツリー・ブロックがスライスにおいて第1の符号化ツリー・ブロックと同じ符号化ツリー・ブロックの行に属することは、ビットストリーム適合性の要件である。

`entropy_coding_sync_enabled_flag`が1に等しく、さらに、スライス・セグメントにおいて第1の符号化ツリー・ブロックがタイルにおいて符号化ツリー・ブロックの行の第1の符号化ツリー・ブロックでないときに、スライス・セグメントにおいて最後の符号化ツリー・ブロックがスライス・セグメントにおいて第1

50

の符号化ツリー・ブロックと同じ符号化ツリー・ブロックの行に属することは、ビットストリーム適合性の要件である。

【0240】

すでに述べたように、CTB30中の符号化／復号化順序は、1つよりも多いタイルが画像に存在する場合、ラスタ方法において第1のタイルをスキャンしそれから次のタイルを訪問して上から下まで行ごとの開始をもたらす。

【0241】

デコーダ5およびしたがってエンコーダは、画像のスライス・セグメント14のエントロピー復号化（符号化）において以下のように働く：

【0242】

A1) 現在復号化／符号化された構文要素synElがタイル50、スライス・セグメント14またはCTBの行の第1の構文要素であるときはいつでも、図29の初期化プロセスが開始される。

A2) そうでなければ、この構文要素の復号化は、現在のエントロピー・コンテキストを用いて起こる。

A3) 現在の構文要素がCTB30において最後の構文要素であった場合、それから、図30に示すようなエントロピー・コンテキスト記憶プロセスが開始される。

A4) そのプロセスは、A1)で次の構文要素を進める。

【0243】

初期化プロセスにおいて、synElがスライス・セグメント14またはタイル50の第1の構文要素であるかどうかについてチェック200される。イエス(yes)の場合、コンテキストは、ステップ202においていかなる前のスライス・セグメントから独立して初期化される。ノー(no)の場合、synElがCTB30の行の第1の構文要素でありさらにentropy_coding_sync_enabled_flagが1に等しいかどうかについてチェック204される。イエス(yes)の場合、等しいタイルのCTB30の前のラインにおいて、第2のCTB30が利用可能であるかどうかについてチェック206される(図23を参照)。イエス(yes)の場合、40に従うコンテキスト適応は、40タイプの適応のための現在格納されたコンテキスト確率を用いてステップ210において実行される。そうでない場合、コンテキストは、ステップ202においていかなる前のスライス・セグメントから独立して初期化される。チェック204がノー(no)を明らかにする場合、synElが依存スライス・セグメント14の第1のCTBにおいて第1の構文要素であるかどうかについて、さらに、dependent_slice_segment_flagが1に等しいかどうかについて、212においてチェックされ、さらに、イエス(yes)の場合、38に従うコンテキスト適応は、38タイプの適応のための現在格納されたコンテキスト確率を用いてステップ214において実行される。ステップ214、212、210および202のいずれかの後に、復号化／符号化が実際に開始される。

【0244】

1に等しいdependent_slice_segment_flagを有する依存スライス・セグメントは、ほとんど符号化効率ペナルティを有しない符号化／復号化遅延をさらに低減するのに役立つ。

【0245】

図30の記憶プロセスにおいて、符号化／復号化されたsynElがCTB30の行の第2のCTB30の最後の構文要素であるかどうかについてさらにentropy_coding_sync_enabled_flagが1に等しいかどうかについてステップ300においてチェックされる。イエス(yes)の場合、現在のエントロピー・コンテキストは、ステップ302において格納され、すなわち、コンテキストのエントロピー符号化確率が40タイプの適応のために特定される記憶において格納される。同様に、符号化／復号化されたsynElがスライス・セグメント14の最後の構文要素であるかどうかについてさらにdependent_slice_segment_flagが1に等

10

20

30

40

50

しいかどうかについて、ステップ300または302に加えて、ステップ304においてチェックされる。イエス (yes) の場合、現在のエントロピー・コンテキストは、ステップ306において格納され、すなわち、コンテキストのエントロピー符号化確率が38タイプの適応のために特定される記憶において格納される。

【0246】

構文要素がCTB行の第1のsynelであるかどうかについて質問するいかなるチェックも、例えばスライス・セグメントのヘッダ内の構文要素slice_address400、すなわち、復号化順序に沿ってそれぞれのスライス・セグメントの開始の位置を明らかにする開始構文要素を利用することに注意されたい。

【0247】

WPP処理を用いてデータストリーム12から画像10を再構成する際に、デコーダは、WPPサブストリーム・エントリ・ポイントを検索するために後の開始構文部分400を正確に利用することができる。それぞれのスライス・セグメントが画像10内でそれぞれのスライス・セグメントの復号化開始の位置を示す開始構文部分400を含むので、デコーダは、スライス・セグメントの開始構文部分400を用いて、画像の左側で開始するスライス・セグメントを識別することによって、スライス・セグメントがグループ化されるWPPサブストリームのエントリ・ポイントを識別することができる。それから、デコーダは、スライス順序に従ってWPPサブストリームの復号化を連続的に開始してWPPサブストリームを千鳥状に並列に復号化することができる。スライス・セグメントは、それらの伝送が全体の伝送エンドツーエンド遅延をさらに低減するためにWPPサブストリーム中でインターリーブされ得るように、1つの画像幅、すなわちCTBの1つの行よりも小さくてもよい。エンコーダは、画像(10)内でそれぞれのスライスの符号化開始の位置を示す開始構文部分(400)を有するそれぞれのスライス(14)を提供し、さらに、WPPサブストリームごとに、スライス順序において第1のスライスが画像の左側で開始するように、スライスをWPPサブストリームにグループ化する。エンコーダは、単独で、画像を符号化する際にWPP処理を用いることができる：エンコーダは、スライス順序に従ってWPPサブストリームの符号化を連続的に開始してWPPサブストリームを千鳥状に並列に符号化する。

【0248】

ところで、WPPサブストリームのエントリ・ポイントの位置を決めるための手段としてスライス・セグメントの開始構文部分を用いる後の態様が、依存スライス概念なしに用いられ得る。

【0249】

それは、以下のように上述の変数をセットすることによって、並列処理画像10のためのすべてに実現可能である：

【0250】

	tiles_enabled_flag==0	tiles_enabled_flag==1	entropy_coding_sync_enable_flag==1	tiles_enabled_flag==1	entropy_coding_sync_enable_flag==1
dependent_slice_segment_flag	0/1	0/1	0/1	0/1	0/1
タイルへの画像再分割	No	Yes	No	Yes	No
WPPサブストリームまたはタイルの開始を示すエントリ・ポイントを有するスライス再分割	No	Yes	Yes	No	No
可能な並列処理	スライスにおいて	タイルにおいて	WPP	タイルにおいて	WPP

【0251】

WPPをタイル・パーティション分割と混合することは、実現可能である。その場合、

1つは、タイルを個々の画像とみなすことができる：WPPを用いるそれぞれは、1つ以上の依存スライス・セグメントを有するスライスからなり、さらに、ステップ300および208におけるチェックは、ステップ204およびA1が現在のタイルのCTB30行において第1のCTB30を参照すると同じように、同じタイルにおいて上述のCTB行において第2のCTBを参照する。その場合、上述のテーブルは、拡張され得る：

【0252】

	tiles_enabled_flag	tiles_enabled_flag==1	entropy_coding_sync_enabled_flag==1	tiles_enabled_flag==1	entropy_coding_sync_enabled_flag==1	entropy_coding_sync_enabled_flag==1	entropy_coding_sync_enabled_flag==1
dependent_slice_segment_flag	0/1	0/1	0/1	0/1	0/1	0/1	0/1
タイルへの画像再分割	No	Yes	No	Yes	No	Yes	Yes
WPPサブストリームまたはタイルの開始を示すエントリ・ポイントを有するスライス再分割	No	Yes	Yes	No	No	No (それ自体の依存スライスにおけるタイルのそれぞれのサブストリーム)	Yes (別々のスライスにおけるそれぞれのタイルおよびタイルのスライス内のWPPサブストリームのためのエントリ・ポイント)
可能な並列処理	スライスにおいて	タイルにおいて	WPP	タイルにおいて	WPP	タイル内のWPPサブストリーム	タイル内のWPPサブストリーム

【0253】

簡潔な注意として、後の拡張は、実施形態2で可能にされている。実施形態2は、以下の処理を可能にする：

【0254】

10

20

30

40

	構文レベル				
tiles_or_entropy_coding_sync_idc	画像ごと	0	1	2	3
cabac_independent_flag	スライスごと	存在しない	存在しない	存在しない	0/1
dependent_slice_flag	スライスごと	0/1	0/1	0/1	0/1
タイルへの画像再分割		No	Yes	No	No
WPPサブストリームまたはタイルの開始を示すエントリ・ポイントを有するスライス再分割		No	Yes	Yes	No
可能な並列処理		スライスにおいて	タイルにおいて	WPP	スライスのエントロピー符号化

10

【0255】

しかし、以下の拡張について、以下のテーブルが生じる：

【0256】

画像パラメータ・セットのセマンティックに加える：

`tiles_or_entropy_coding_sync_idc`が4に等しい場合、それぞれの、しかし、CTBの第1の行は、1にセットされる依存スライス・フラグを有する異なるスライスに含まれる。異なる行のCTBは、同じスライスに存在してはならない。CTB列ごとに1つよりも多いスライスが存在してもよい。

20

【0257】

`tiles_or_entropy_coding_sync_idc`が5に等しい場合、それぞれのCTB、しかし、第1のタイルは、異なるスライスに含まれなければならない。異なるタイルのCTBは、同じスライスに存在してはならない。タイルごとに存在する1つよりも多いスライスが存在してもよい。

【0258】

さらなる説明のために、図31を参照する。

【0259】

すなわち、上述のテーブルは、拡張され得る：

30

【0260】

	構文レベル								
tiles_or_entropy_coding_sync_idc	画像ごと	0	1	2	3	5	4	6	7
cabac_independent_flag	スライスごと	存在しない	存在しない	存在しない	0/1	存在しない	存在しない	存在しない	存在しない
dependent_slice_flag	スライスごと	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1
タイルへの画像再分割		No	Yes	No	No	Yes	No	Yes	Yes
WPPサブストリームまたはタイルの開始を示すエントロピー・ポイントを有するスライス再分割		No	Yes	Yes	No	No	No	No (それ自体の依存スライスにおけるタイルのそれぞれのサブストリーム)	Yes (個々のスライスにおけるそれぞれのタイルおよびタイルのスライス内のWPPサブストリームのためのエントロピー・ポイント)
可能な並列処理		スライスにおいて	スライスにおいて	WPP	スライスエントロピー符号化	タイルにおいて	WPP	タイル内のWPPサブストリーム	タイル内のWPPサブストリーム

【0261】

上述の実施形態に関して、デコーダが、例えば、第1および第2のモードにおいて、tiles_or_entropy_coding_sync_idc=1、2に回答して、並列サブセクションへの現在のスライスの再分割を明らかにする現在のスライスから情報を読み取るように構成され得るべきであることに注意すべきであり、並列サブセクションは、第1の並列サブセクションの最後にコンテキスト適応エントロピー復号化を中止し、さらに、第1のモードにおいて、前の並列サブセクションのシンボル確率の保存された状態に応じたシンボル確率の初期化と、第2のモードにおいて、いかなる前に復号化されたスライスおよびいかなる前に復号化された並列サブセクションから独立したシンボル確率の初期化とを含むいかなる後の並列サブセクションの開始時に新たにコンテキスト適応エントロピー復号化を再開する、WPPサブストリームまたはタイルであり得る。

【0262】

このように、上述の説明は、例えばタイル、波面並列処理(WPP)サブストリーム、スライスまたはエントロピー・スライスにおいて構成されるように、新しいHEVC符号化標準によって提供されるような構成されたビデオ・データの低遅延符号化、復号化、カプセル化および伝送のための方法を明らかにしている。

【0263】

特に、どのように符号化、復号化および伝送プロセスにおいて最小待ち時間を得るため

に会話のシナリオにおいて並列符号化されたデータを転送するかについて定義されている。したがって、ゲーム、遠隔手術などのような最小の遅延アプリケーションを可能にするために、パイプライン並列符号化、伝送および復号化アプローチが記載されている。

【0264】

さらに、上述の実施形態は、それを低遅延伝送シナリオにおいて用いることができるようにするために、波面並列処理(WPP)のギャップを埋めている。したがって、WPPサブストリーム0のための新しいカプセル化フォーマットは、依存スライスが示されている。この依存スライスは、エントロピー・スライス・データ、WPPサブストリーム、LCUの完全な行、ちょうどスライスのフラグメントを含むことができ、従来の送信されたスライス・ヘッダも、含まれたフラグメント・データに適用される。含まれたデータは、サブ・スライス・ヘッダにおいて信号伝達される。

10

【0265】

新しいスライスのためのネーミングが「サブセット／軽量スライス」でもあり得るが、ネーム「依存スライス」がより良好であることが見出されていることに最後に注意されたい。

【0266】

符号化および転送において並列化のレベルを記載する信号伝達を示されている。

【0267】

いくつかの態様が装置との関連で記載されているにもかかわらず、これらの態様は、対応する方法の説明も表すことも明らかであり、ブロックまたは装置は、方法ステップまたは方法ステップの特徴に対応する。同様に、方法ステップとの関連で記載されている態様は、対応するブロック若しくはアイテムまたは対応する装置の特徴の説明も表す。方法ステップのいくらかまたはすべては、例えば、マイクロプロセッサ、プログラム可能なコンピュータまたは電子回路のようなハードウェア装置によって（またはそれを用いて）実行されてもよい。いくつかの実施形態において、最も重要な方法ステップのいずれかの1つ以上は、そのような装置によって実行されてもよい。

20

【0268】

特定の実施要件に応じて、本発明の実施形態は、ハードウェアにおいてまたはソフトウェアにおいて実施することができる。実施は、それぞれの方法が実行されるように、プログラム可能なコンピュータシステムと協働する（または協働することができる）電子的に可読の制御信号が格納される、デジタル記憶媒体、例えばフロッピー（登録商標）ディスク、DVD、ブルーレイ（登録商標）、CD、ROM、PROM、EPROM、EEPROMまたはFLASHメモリを用いて実行することができる。したがって、デジタル記憶媒体は、コンピュータ可読であってもよい。

30

【0269】

本発明によるいくつかの実施形態は、ここに記載される方法のうちの1つが実行されるように、プログラム可能なコンピュータシステムと協働することができる電子的に可読の制御信号を有するデータキャリアを含む。

【0270】

一般に、本発明の実施形態は、プログラムコードを有するコンピュータプログラム製品として実施することができ、そのプログラムコードは、コンピュータプログラム製品がコンピュータ上で実行されるときに、それらの方法のうちの1つを実行するために働く。プログラムコードは、例えば、機械可読のキャリアに格納されてもよい。

40

【0271】

他の実施形態は、機械可読のキャリアに格納される、ここに記載される方法のうちの1つを実行するためのコンピュータプログラムを含む。

【0272】

したがって、換言すれば、本発明の方法の実施形態は、コンピュータプログラムがコンピュータ上で実行されるときに、ここに記載される方法のうちの1つを実行するためのプログラムコードを有するコンピュータプログラムである。

50

【0273】

したがって、本発明の方法のさらなる実施形態は、それに記録される、ここに記載される方法のうちの1つを実行するためのコンピュータプログラムを含むデータキャリア（またはデジタル記憶媒体またはコンピュータ可読の媒体）である。データキャリア、デジタル記憶媒体または記録媒体は、典型的に有形でありおよび／または一時的でない。

【0274】

したがって、本発明の方法のさらなる実施形態は、ここに記載される方法のうちの1つを実行するためのコンピュータプログラムを表すデータストリームまたは一連の信号である。データストリームまたは一連の信号は、例えば、データ通信接続を介して、例えばインターネットを介して、転送されるように構成されてもよい。

10

【0275】

さらなる実施形態は、ここに記載される方法のうちの1つを実行するように構成されまたは適している処理手段、例えばコンピュータまたはプログラム可能な論理デバイスを含む。

【0276】

さらなる実施形態は、ここに記載される方法のうちの1つを実行するためのコンピュータプログラムがインストールされているコンピュータを含む。

【0277】

本発明によるさらなる実施形態は、ここに記載される方法のうちの1つを実行するためのコンピュータプログラムをレシーバに（例えば、電子的にまたは光学的に）転送するように構成される装置またはシステムを含む。レシーバは、例えば、コンピュータ、モバイルデバイス、メモリデバイスなどであってもよい。装置またはシステムは、例えば、コンピュータプログラムをレシーバに転送するためのファイルサーバを含んでもよい。

20

【0278】

いくつかの実施形態において、プログラム可能な論理デバイス（例えばフィールドプログラム可能なゲートアレイ）は、ここに記載される方法の機能のいくらかまたはすべてを実行するために用いられてもよい。いくつかの実施形態において、フィールドプログラム可能なゲートアレイは、ここに記載される方法のうちの1つを実行するために、マイクロプロセッサと協働してもよい。一般に、その方法は、好ましくは、いかなるハードウェア装置によっても実行される。

30

【0279】

上述の実施形態は、本発明の原理のために単に例示するだけである。ここに記載される構成および詳細の修正および変更が他の当業者にとって明らかであるものと理解される。したがって、本発明は、特許請求の範囲によってだけ制限され、ここに実施形態の記述および説明として示される具体的な詳細によって制限されないと意図される。

【0280】

文献

[1] Thomas Wiegand, Gary J. Sullivan, Gisle Bjontegaard, Ajay Luthra, "Overview of the H.264/AVC Video Coding Standard", IEEE Trans. Circuits Syst. Video Technol., vol. 13, N7, July 2003.

40

[2] JCT-VC, "High-Efficiency Video Coding (HEVC) text specification Working Draft 6", JCTVC-H1003, February 2012.

[3] ISO/IEC 13818-1: MPEG-2 Systems specification.

【図 1】

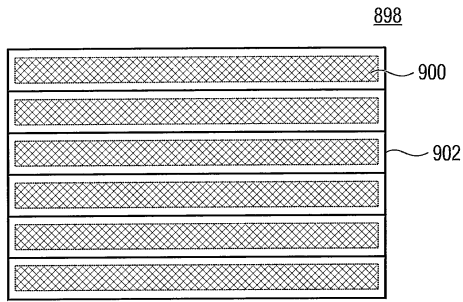


FIG 1

【図 2】

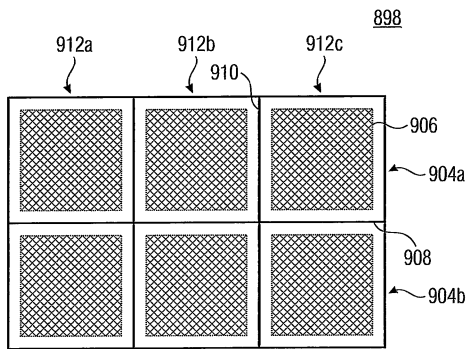


FIG 2

【図 3】

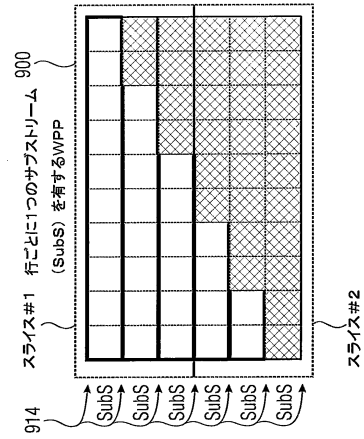
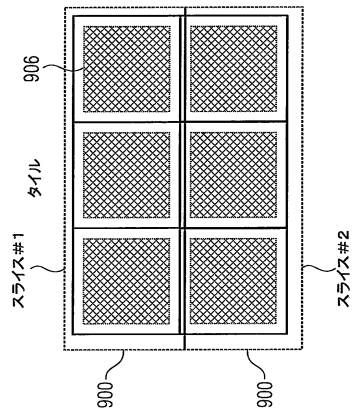


FIG 3



【図 4】

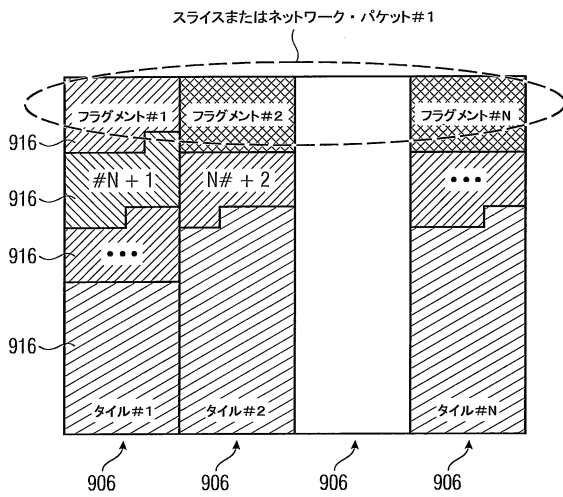


FIG 4

【図 5】

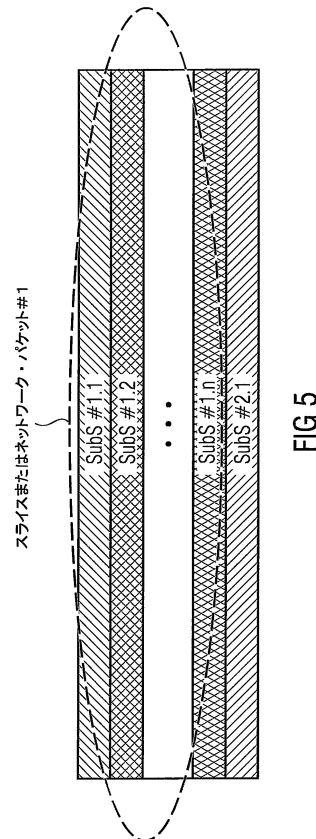


FIG 5

【図 6】

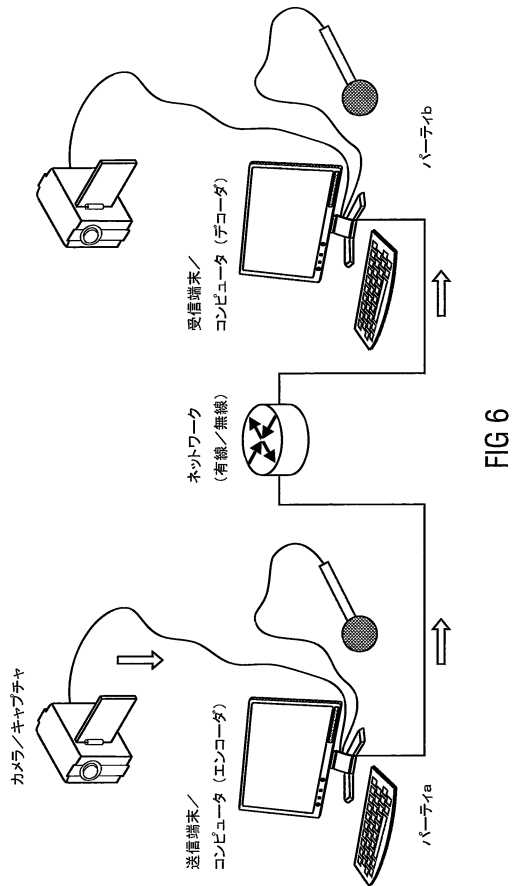


FIG 6

【図 7】

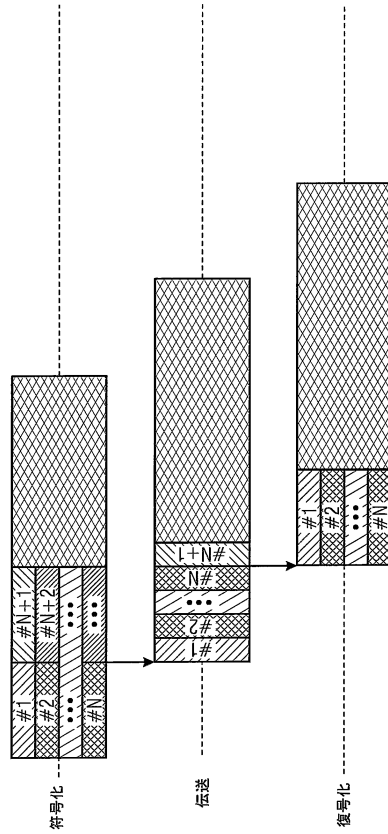


FIG 7

【図 8】

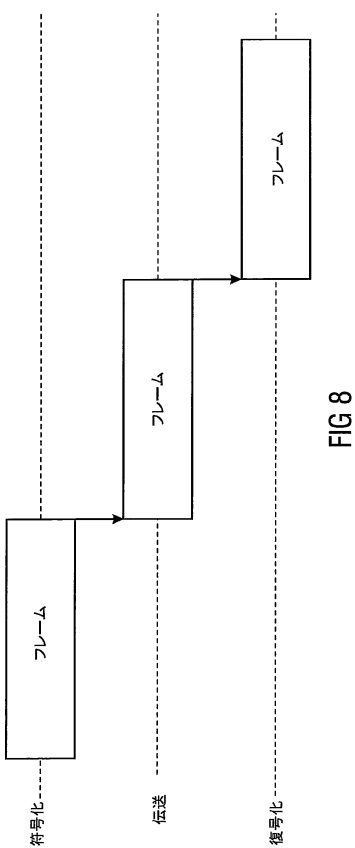


FIG 8

【図 9】

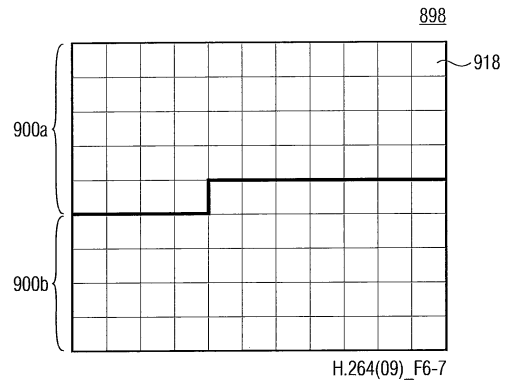


FIG 9

【図 10】

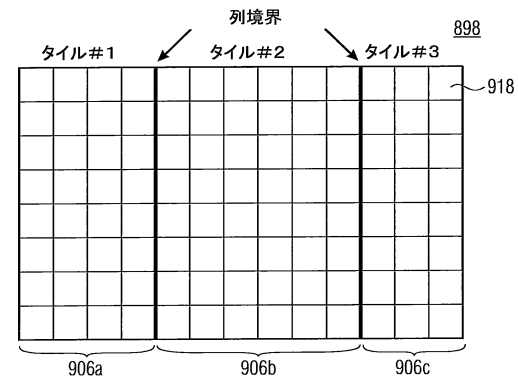


FIG 10

【図 11 a】

seq_parameter_set_rbsp() {	記述子
profile_idc	u(8)
reserved_zero_8bits /* equal to 0 */	u(8)
level_idc	u(8)
seq_parameter_set_id	ue(v)
chroma_format_idc	ue(v)
if(chroma_format_idc == 3)	
separate_colour_plane_flag	u(1)
max_temporal_layers_minus1	u(3)
pic_width_in_luma_samples	ue(v)
pic_height_in_luma_samples	ue(v)
pic_cropping_flag	u(1)
if(pic_cropping_flag) {	
pic_crop_left_offset	ue(v)
pic_crop_right_offset	ue(v)
pic_crop_top_offset	ue(v)
pic_crop_bottom_offset	ue(v)
}	
bit_depth_luma_minus8	ue(v)
bit_depth_chroma_minus8	ue(v)
pcm_enabled_flag	u(1)
if(pcm_enabled_flag) {	
pcm_bit_depth_luma_minus1	u(4)
pcm_bit_depth_chroma_minus1	u(4)
}	
qpprime_y_zero_transquant_bypass_flag	u(1)
log2_max_pic_order_cnt_lsb_minus4	ue(v)
for(i = 0; i <= max_temporal_layers_minus1; i++) {	
max_dec_pic_buffering[i]	ue(v)

FIG 11A

FIG 11

FIG 11A
FIG 11B
FIG 11C

【図 11 b】

num_reorder_pics[i]	ue(v)
max_latency_increase[i]	ue(v)
}	
restricted_ref_pic_lists_flag	u(1)
if(restricted_ref_pic_lists_flag)	
lists_modification_present_flag	u(1)
log2_min_coding_block_size_minus3	ue(v)
log2_diff_max_min_coding_block_size	ue(v)
log2_min_transform_block_size_minus2	ue(v)
log2_diff_max_min_transform_block_size	ue(v)
if(pcm_enabled_flag) {	
log2_min_pcm_coding_block_size_minus3	ue(v)
log2_diff_max_min_pcm_coding_block_size	ue(v)
}	
max_transform_hierarchy_depth_inter	ue(v)
max_transform_hierarchy_depth_intra	ue(v)
scaling_list_enable_flag	
chroma_pred_from_luma_enabled_flag	u(1)
deblocking_filter_in_aps_enabled_flag	u(1)
seq_loop_filter_across_slices_enabled_flag	u(1)
asymmetric_motion_partitions_enabled_flag	u(1)
non_square_quadtree_enabled_flag	u(1)
sample_adaptive_offset_enabled_flag	u(1)
adaptive_loop_filter_enabled_flag	u(1)
if(adaptive_loop_filter_enabled_flag)	
alf_coef_in_slice_flag	u(1)
if(pcm_enabled_flag)	
pcm_loop_filter_disable_flag	u(1)
temporal_id_nesting_flag	u(1)

FIG 11B

FIG 11

FIG 11A
FIG 11B
FIG 11C

【図 11 c】

if(log2_min_coding_block_size_minus3 == 0)	
inter_4x4_enabled_flag	u(1)
num_short_term_ref_pic_sets	ue(v)
for(i = 0; i < num_short_term_ref_pic_sets; i++)	
short_term_ref_pic_set[i]	
long_term_ref_pics_present_flag	u(1)
tiles_or_entropy_coding_sync_idc	u(2)
if(tiles_or_entropy_coding_sync_idc == 1) {	
num_tile_columns_minus1	ue(v)
num_tile_rows_minus1	ue(v)
uniform_spacing_flag	u(1)
if(!uniform_spacing_flag) {	
for(i = 0; i < num_tile_columns_minus1; i++)	
column_width[i]	ue(v)
for(i = 0; i < num_tile_rows_minus1; i++)	
row_height[i]	ue(v)
}	
loop_filter_across_tiles_enabled_flag	u(1)
}	
vui_parameters_present_flag	u(1)
if(vui_parameters_present_flag)	
vui_parameters()	
sps_extension_flag	u(1)
if(sps_extension_flag)	
while(more_rbsp_data())	
sps_extension_data_flag	u(1)
rbsp_trailing_bits()	
}	

FIG 11C

FIG 11

FIG 11A
FIG 11B
FIG 11C

【図 12 a】

pic_parameter_set_rbsp() {	記述子
pic_parameter_set_id	ue(v)
seq_parameter_set_id	ue(v)
sign_data_hiding_flag	u(1)
if(sign_data_hiding_flag)	
sign_hiding_threshold	u(4)
cabac_init_present_flag	u(1)
num_ref_idx_l0_default_active_minus1	ue(v)
num_ref_idx_l1_default_active_minus1	ue(v)
pic_init_qp_minus26	se(v)
constrained_intra_pred_flag	u(1)
enable_temporal_mvp_flag	u(1)
slice_granularity	u(2)
max_cu_qp_delta_depth	ue(v)
cb_qp_offset	se(v)
cr_qp_offset	se(v)
weighted_pred_flag	u(1)
weighted_bipred_idc	u(2)
output_flag_present_flag	u(1)
if(tiles_or_entropy_coding_sync_idc == 1) {	
tile_info_present_flag	u(1)
tile_control_present_flag	u(1)
if(tile_info_present_flag) {	
num_tile_columns_minus1	ue(v)

FIG 12A

FIG 12

FIG 12A
FIG 12B

【図 1 2 b】

num_tile_rows_minus1	ue(v)
uniform_spacing_flag	u(1)
if(uniform_spacing_flag) {	
for(i = 0; i < num_tile_columns_minus1; i++)	
column_width[i]	ue(v)
for(i = 0; i < num_tile_rows_minus1; i++)	
row_height[i]	ue(v)
}	
}	
if(tile_control_present_flag)	
loop_filter_across_tiles_enabled_flag	u(1)
} else if(tiles_or_entropy_coding_sync_idc == 2)	
num_substreams_minus1	ue(v)
deblocking_filter_control_present_flag	u(1)
if(slice_type == P slice_type == B)	
log2_parallel_merge_level_minus2	ue(v)
pps_extension_flag	u(1)
if(pps_extension_flag)	
while(more_rbsp_data())	
pps_extension_data_flag	u(1)
rbps_trailing_bits()	
}	

FIG 12B

FIG 12	FIG 12A
	FIG 12B

【図 1 3 a】

slice_header() {	記述子
first_slice_in_pic_flag	u(1)
if(first_slice_in_pic_flag == 0)	
slice_address	u(v)
slice_type	ue(v)
entropy_slice_flag	u(1)
if(!entropy_slice_flag) {	
pic_parameter_set_id	ue(v)
if(!output_flag_present_flag)	
pic_output_flag	u(1)
if(!separate_colour_plane_flag == 1)	
colour_plane_id	u(2)
if(!idrPicFlag) {	
idr_pic_id	ue(v)
no_output_of_prior_pics_flag	u(1)
} else {	
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	
short_term_ref_pic_set_idx	u(v)
if(!long_term_ref_pics_present_flag) {	
num_long_term_pics	ue(v)
for(i = 0; i < num_long_term_pics; i++) {	
delta_poc_lsb_lf[i]	ue(v)
delta_poc_msb_present_flag[i]	u(1)
if(!delta_poc_msb_present_flag[i])	
delta_poc_msb_cycle_lf_minus1[i]	ue(v)
used_by_curr_pic_lf_flag[i]	u(1)
}	
}	
if(!sample_adaptive_offset_enabled_flag) {	
slice_sao_interleaving_flag	u(1)
slice_sample_adaptive_offset_flag	u(1)

FIG 13A

FIG 13	FIG 13A
	FIG 13B
	FIG 13C

【図 1 3 b】

if(slice_sao_interleaving_flag && slice_sample_adaptive_offset_flag) {	
sao_cb_enable_flag	u(1)
sao_cr_enable_flag	u(1)
}	
if(scaling_list_enable_flag deblocking_filter_in_aps_enabled_flag (sample_adaptive_offset_enabled_flag && !slice_sao_interleaving_flag) adaptive_loop_filter_enabled_flag)	
aps_id	ue(v)
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) {	
ref_pic_list_modification()	
ref_pic_list_combination()	
}	
if(slice_type == B)	
mvd_l1_zero_flag	u(1)
}	
if(cabac_init_present_flag && slice_type != 1)	
cabac_init_flag	u(1)
if(!entropy_slice_flag) {	
slice_qp_delta	se(v)
if(deblocking_filter_control_present_flag) {	
if(deblocking_filter_in_aps_enabled_flag)	
inherit_dbl_params_from_aps_flag	u(1)
if(!inherit_dbl_params_from_aps_flag) {	
disable_deblocking_filter_flag	u(1)
if(!disable_deblocking_filter_flag) {	
beta_offset_div2	se(v)

FIG 13B

FIG 13	FIG 13A
	FIG 13B
	FIG 13C

【図 1 3 c】

tc_offset_div2	se(v)
}	
}	
if(slice_type == B)	
collocated_from_l0_flag	u(1)
if(slice_type != 1 && ((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_idc == 1 && slice_type == B))	
pred_weight_table()	
}	
if(slice_type == P slice_type == B)	
five_minus_max_num_merge_cand	ue(v)
if(adaptive_loop_filter_enabled_flag) {	
slice_adaptive_loop_filter_flag	u(1)
if(slice_adaptive_loop_filter_flag && alf_coef_in_slice_flag)	
alf_param()	
if(slice_adaptive_loop_filter_flag && alf_coef_in_slice_flag)	
alf_cu_control_param()	
}	
if(seq_loop_filter_across_slices_enabled_flag && (slice_adaptive_loop_filter_flag slice_sample_adaptive_offset_flag !disable_deblocking_filter_flag))	
slice_loop_filter_across_slices_enabled_flag	u(1)
if(tiles_or_entropy_coding_sync_idc > 0) {	
num_entry_point_offsets	ue(v)
if(num_entry_point_offsets > 0) {	
offset_len_minus1	ue(v)
for(i = 0; i < num_entry_point_offsets; i++)	
entry_point_offset[i]	u(v)
}	
}	

FIG 13C

FIG 13	FIG 13A
	FIG 13B
	FIG 13C

【図 14】

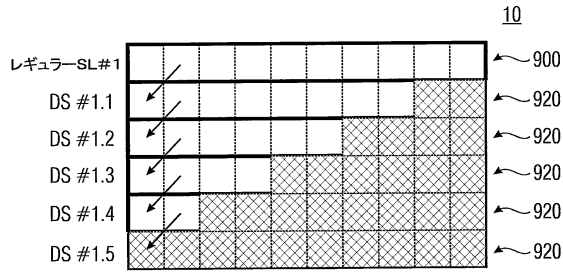


FIG 14

【図 15】

pic_parameter_set_rbsp() {	記述子
...	
dependent_slices_present_flag	u(1)
last_ctb_cabac_init_flg	u(1)
}	

FIG 15

【図 16】

slice_header() {	記述子
first_slice_in_pic_flag	u(1)
if (first_slice_in_pic_flag == 0)	
slice_address	u(v)
slice_type	ue(v)
entropy_slice_flag	u(1)
dependent_slice_flag	u(1)
if (entropy_slice_flag) {	
if (dependent_slice_flag) {	
...	
} else {	
no_cabac_reset_flag	u(1)
}	
...	
if (cabac_init_present_flag & slice_type != I & !no_cabac_reset_flag)	
cabac_init_flag	u(1)
...	
if (slice_type == P slice_type == B)	
five_minus_max_num_merge_cand	ue(v)
if (adaptive_loop_filter_enabled_flag) {	
slice_adaptive_loop_filter_flag	u(1)
if (slice_adaptive_loop_filter_flag & alf_coef_in_slice_flag)	
alf_param()	
if (slice_adaptive_loop_filter_flag & alf_coef_in_slice_flag)	
alf_cu_control_param()	
}	
if (seq_loop_filter_across_slices_enabled_flag && (slice_adaptive_loop_filter_flag slice_sample_adaptive_offset_flag disable_deblocking_filter_flag))	
slice_loop_filter_across_slices_enabled_flag	u(1)
if (tiles_or_entropy_coding_sync_idc >= 0) {	
num_entry_point_offsets	ue(v)
if (num_entry_point_offsets > 0) {	
offset_len_minus1	ue(v)
for (i = 0; i < num_entry_point_offsets; i++)	
entry_point_offset[i]	u(v)
}	
}	
}	

FIG 16

【図 17】

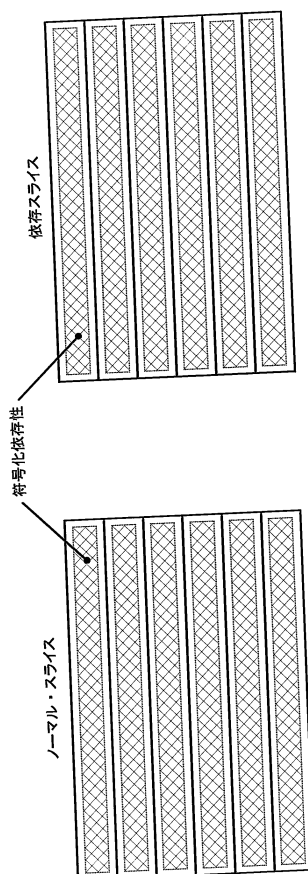


FIG 17

【図 18】

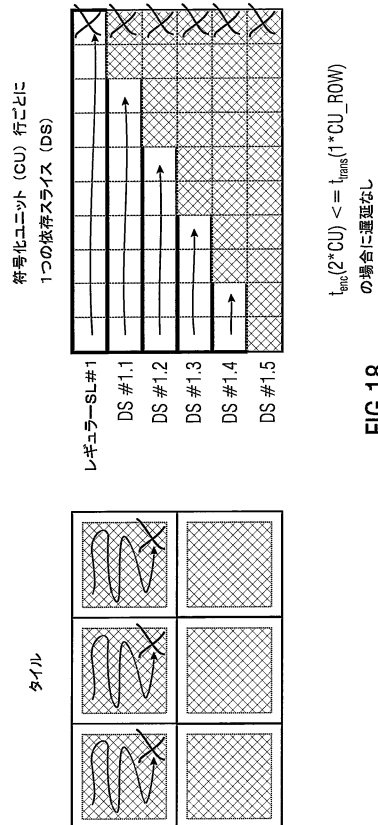


FIG 18

【図 19】

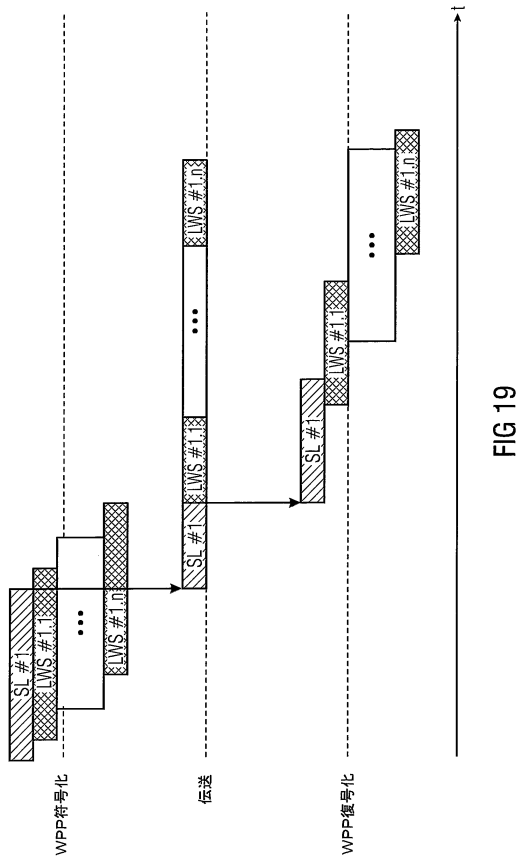


FIG 19

【図 20】

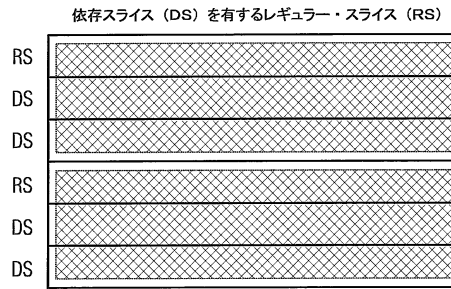


FIG 20

【図 21】

slice_header() {	記述子
first_slice_in_pic_flag	u(1)
pic_parameter_set_id	ue(v)
if(first_slice_in_pic_flag)	
slice_address	u(v)
if(dependent_slice_enabled_flag && !first_slice_in_pic_flag)	
dependent_slice_flag	u(1)
if(!dependent_slice_flag) {	
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(RapPicFlag) {	
rap_pic_id	ue(v)
no_output_of_prior_pics_flag	u(1)
}	
if(!IdrPicFlag) {	
if(tiles_or_entropy_coding_sync_idc == 1 tiles_or_entropy_coding_sync_idc == 2) {	
num_entry_point_offsets	ue(v)
if(num_entry_point_offsets > 0) {	
offset_len_minus1	ue(v)
for(i = 0; i < num_entry_point_offsets; i++)	
entry_point_offset[i]	u(v)
}	
}	
if(slice_header_extension_present_flag) {	
slice_header_extension_length	ue(v)
for(i = 0; i < slice_header_extension_length; i++)	
slice_header_extension_data_byte	u(8)
}	
byte_alignment()	
}	

FIG 21

【図 22】

pic_parameter_set_rbsp() {	記述子
pic_parameter_set_id	ue(v)
seq_parameter_set_id	ue(v)
sign_data_hiding_flag	u(1)
if(sign_data_hiding_flag)	
sign_hiding_threshold	u(4)
cabac_init_present_flag	u(1)
num_ref_idx_l0_default_active_minus1	ue(v)
num_ref_idx_l1_default_active_minus1	ue(v)
[Ed. (BB): not present in HM software]	
pic_init_qp_minus26	se(v)
constrained_intra_pred_flag	u(1)
slice_granularity	u(2)
diff_cu_qp_delta_depth	ue(v)
cb_qp_offset	se(v)
cr_qp_offset	se(v)
weighted_pred_flag	u(1)
weighted_bipred_idc	u(2)
output_flag_present_flag	u(1)
transquant_bypass_enable_flag	u(1)
dependent_slice_enabled_flag	u(1)
tiles_or_entropy_coding_sync_idc	u(2)
if(tiles_or_entropy_coding_sync_idc == 1) {	
num_tile_columns_minus1	ue(v)
num_tile_rows_minus1	ue(v)
uniform_spacing_flag	u(1)
if(!uniform_spacing_flag) {	
for(i = 0; i < num_tile_columns_minus1; i++)	
column_width[i]	ue(v)
for(i = 0; i < num_tile_rows_minus1; i++)	
row_height[i]	ue(v)
}	
loop_filter_across_tiles_enabled_flag	u(1)
} else if(tiles_or_entropy_coding_sync_idc == 3)	
cabac_independent_flag	u(1)
...	
}	

FIG 22

【图 2 3】

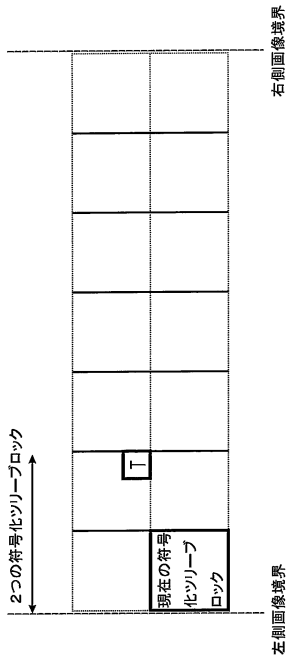


FIG 23

【図 24】

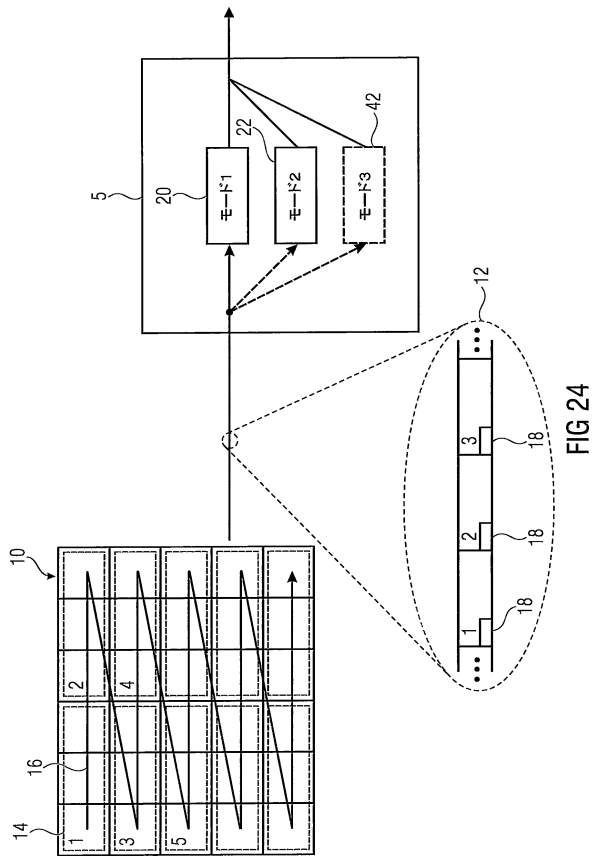


FIG 24

【図 25】

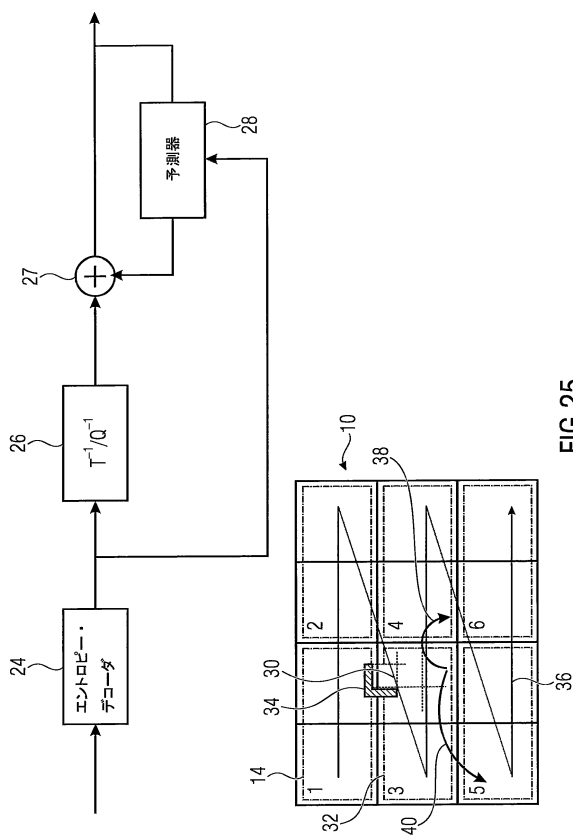


FIG 25

【図 2 6】

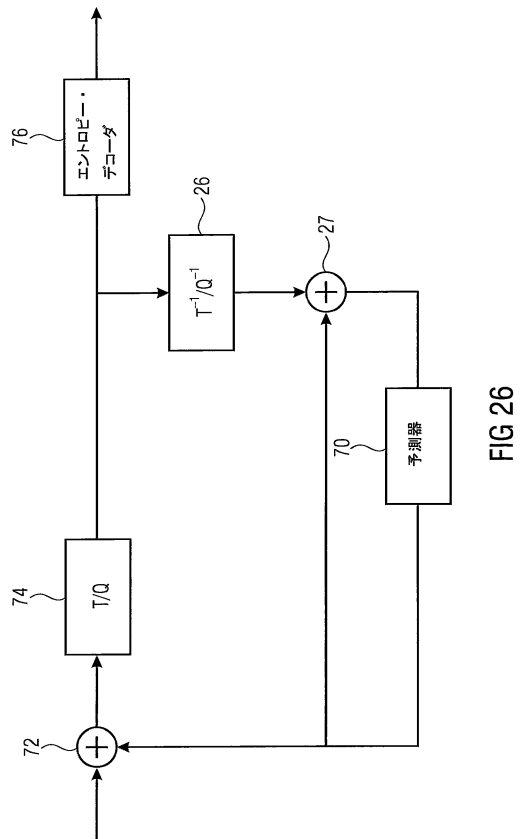


FIG 26

【図 27】

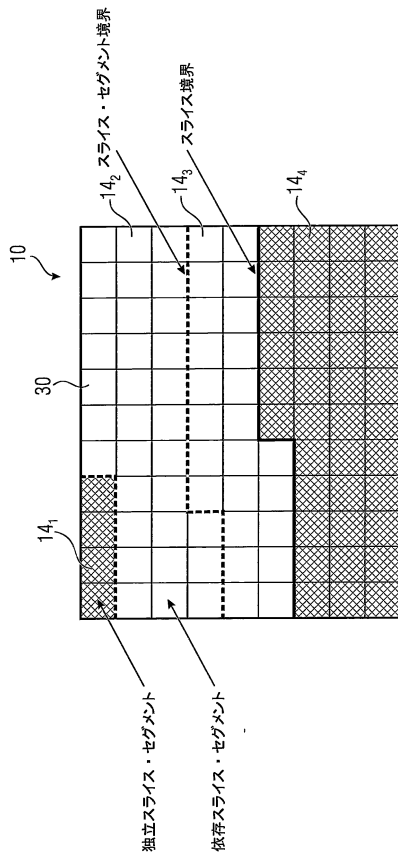


FIG 27

【図 28】

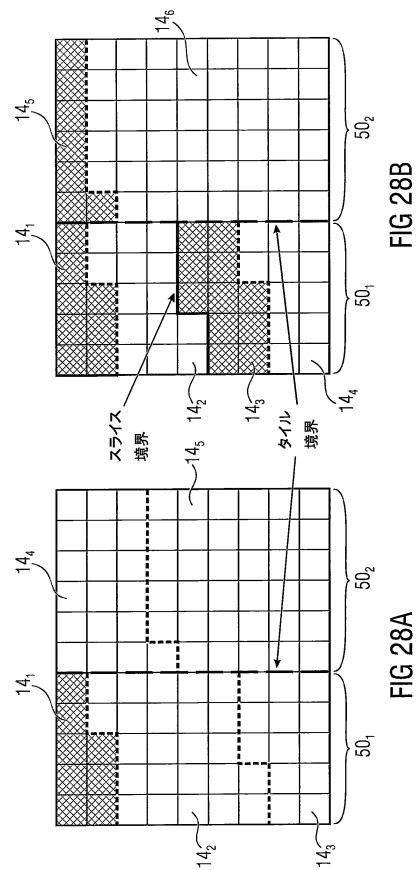


FIG 28B

FIG 28A

【図 29】

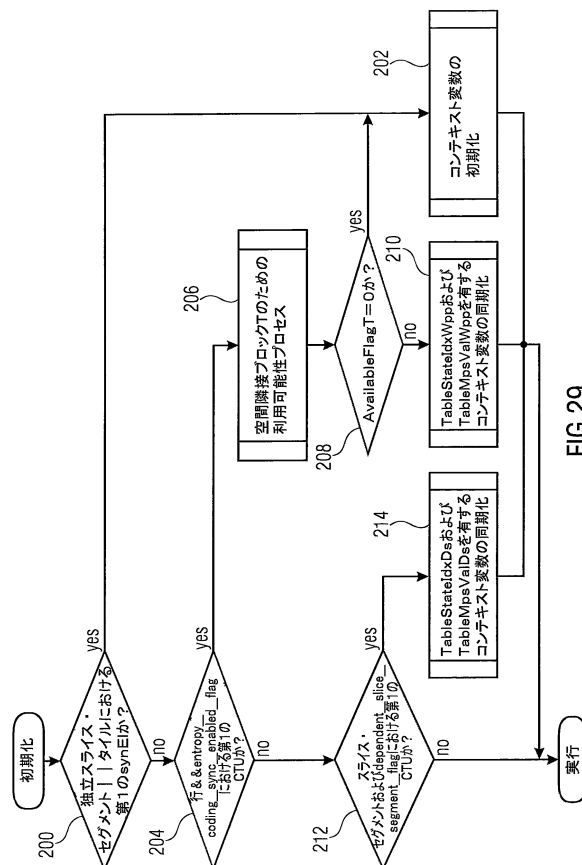


FIG 29

【図 30】

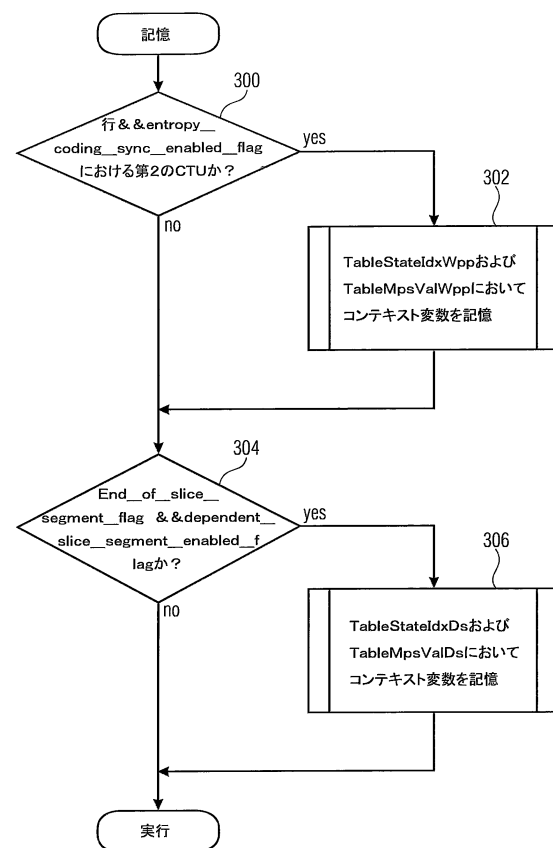


FIG 30

【図 31】

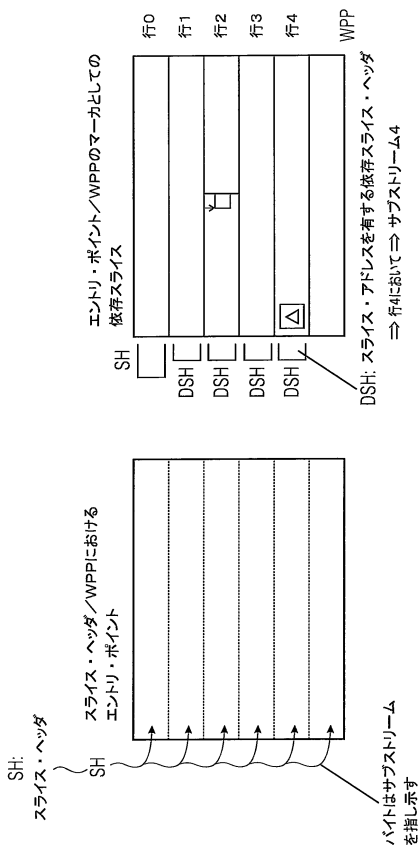


FIG 31

フロントページの続き

- (72)発明者 ヘンケル アナスタージア
ドイツ連邦共和国 1 0 2 4 5 ベルリン ファレンファイトシュトラッセ 8
- (72)発明者 マルペ デトレフ
ドイツ連邦共和国 1 2 1 6 1 ベルリン ズュートヴェストコルゾ 7 0
- (72)発明者 グルーエネベルク カルステン
ドイツ連邦共和国 1 3 5 9 9 ベルリン アディッケスシュトラッセ 4 3
- (72)発明者 スクーピン ローベルト
ドイツ連邦共和国 1 0 5 5 5 ベルリン シュレースヴィガー ウーファー 5

審査官 岩井 健二

(56)参考文献 特開2 0 0 9－1 7 7 7 8 7 (J P, A)

Gordon Clare et al., Wavefront Parallel Processing for HEVC Encoding and Decoding, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 6th Meeting: Torino, IT, 2 0 1 1年 7月, JCTVC-F274, pp.1-16

Gordon Clare et al., Wavefront and Cabac Flush: Different Degrees of Parallelism Without Transcoding, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 6th Meeting: Torino, IT, 2 0 1 1年 7月, JCTVC-F275, pp.1-11

Kiran Misra and Andrew Segall, Harmonization of Entry Points for Tiles and Wavefront Processing, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 7th Meeting: Geneva, CH, 2 0 1 1年1 1月, JCTVC-G722-r1, pp.1-4

Muhammed Coban et al., On Entropy Slices, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 8th Meeting: San Jose, CA, USA, 2 0 1 2年 1月, JCTVC-H0515r2, pp.1-7

Hendry et al., AHG4: Harmonized Method for Signalling Entry Points of tiles and WPP Substreams, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 8th Meeting: San Jose, CA, USA, 2 0 1 2年 2月, JCTVC-H0566 (JCTVC-H0556_r1.docx), pp.1-7

T. Schierl et al., Dependent Slices, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 9th Meeting: Geneva, CH, 2 0 1 2年 4月, JCTVC-I0229, pp.1-7

T. Schierl et al., Dependent slices support in HEVC main profile, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 10th Meeting: Stockholm, SE, 2 0 1 2年 7月, JCTVC-J0264, pp.1-4

(58)調査した分野(Int.Cl., DB名)

H 0 4 N 1 9 / 0 0 - 1 9 / 9 8