(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5041060号
(P5041060)
(45)発行日　平成24年10月3日(2012.10.3)　(24)登録日　平成24年7月20日(2012.7.20)

(51)Int.Cl.　F I
H04N 7/30 (2006.01)　H04N 7/133 Z

請求項の数 2　(全 43 頁)

(21)出願番号　特願2010-290297(P2010-290297)
(22)出願日　平成22年12月27日(2010.12.27)
(62)分割の表示　特願2007-241628(P2007-241628)の分割
原出願日　平成14年10月10日(2002.10.10)
(65)公開番号　特開2011-61877(P2011-61877A)
(43)公開日　平成23年3月24日(2011.3.24)
審査請求日　平成23年1月24日(2011.1.24)

早期審査対象出願

(73)特許権者　000002185
ソニー株式会社
東京都港区港南１丁目７番１号
(74)代理人　100082740
弁理士　田辺　恵基
(72)発明者　矢ケ崎　陽一
東京都港区港南１丁目７番１号ソニー株式会社内
(72)発明者　春原　修
東京都港区港南１丁目７番１号ソニー株式会社内
(72)発明者　村山　淳
東京都港区港南１丁目７番１号ソニー株式会社内

最終頁に続く

(54)【発明の名称】符号化装置及び符号化方法

(57)【特許請求の範囲】
【請求項１】
画像データを符号化処理する符号化装置において、
画像データに対してコンテキストを用いた算術符号化処理を行う符号化手段と、
前記画像データを符号化処理する際の単位であるブロックを非圧縮データのブロックとする場合に、コンテキストを用いた算術符号化処理の終端処理を行うように、前記符号化手段を制御する制御手段と、
を備える符号化装置。
【請求項２】
画像データを符号化処理する符号化方法において、
画像データに対してコンテキストを用いた算術符号化処理を行う符号化ステップと、
前記画像データを符号化処理する際の単位であるブロックを非圧縮データのブロックとする場合に、コンテキストを用いた算術符号化処理の終端処理を行うように、前記符号化ステップの符号化処理を制御する制御ステップと、
を含む符号化方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、ＪＶＴ(ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC)などの様に、離散コサイン変換若しくはカルーネン・レーベ変換等の直交変換と動き補償とによって圧縮された

画像情報（ビットストリーム）を、衛星放送、ケーブルＴＶ若しくはインターネット等のネットワークメディアを介して受信する際に、又は光ディスク、磁気ディスク若しくはフラッシュメモリ等の記憶メディア上で処理する際に用いられる符号化装置及び符号化方法に関するものである。

【背景技術】

【０００２】

近年、画像情報をデジタルとして取り扱い、その際、効率の高い情報の伝送、蓄積を目的とし、画像情報特有の冗長性を利用して、離散コサイン変換等の直交変換と動き補償により圧縮するＭＰＥＧなどの方式に準拠した装置が、放送局などの情報配信、及び一般家庭における情報受信の双方において普及しつつある。

【０００３】

特に、ＭＰＥＧ２（ISO/IEC 13818-2）は、汎用画像符号化方式として定義されており、飛び越し走査画像及び順次走査画像の双方、並びに標準解像度画像及び高精細画像を網羅する標準で、プロフェッショナル用途及びコンシューマー用途の広範なアプリケーションに現在広く用いられている。ＭＰＥＧ２圧縮方式を用いることにより、例えば７２０×４８０画素を持つ標準解像度の飛び越し走査画像であれば４～８Ｍｂｐｓ、１９２０×１０８８画素を持つ高解像度の飛び越し走査画像であれば１８～２２Ｍｂｐｓの符号量（ビットレート）を割り当てることで、高い圧縮率と良好な画質の実現が可能である。

【０００４】

ＭＰＥＧ２は主として放送用に適合する高画質符号化を対象としていたが、ＭＰＥＧ１より低い符号量（ビットレート）、つまりより高い圧縮率の符号化方式には対応していなかった。しかし、携帯端末の普及により、今後そのような符号化方式のニーズは高まると思われ、これに対応してＭＰＥＧ４符号化方式の標準化が行われた。画像符号化方式に関しては、１９９８年１２月にISO/IEC 14496-2としてその規格が国際標準に承認された。

【０００５】

さらに、近年、テレビ会議用の画像符号化を当初の目的として、ＪＶＴ(ITU-T Rec. H.264 ｜ ISO/IEC 14496－10 AVC)という標準の規格化が進んでいる。ＪＶＴはＭＰＥＧ２やＭＰＥＧ４といった従来の符号化方式に比べ、その符号化、復号により多くの演算量が要求されるものの、より高い符号化効率が実現されることが知られている。

【０００６】

ここで、ＭＰＥＧ２やＪＶＴで採用されている、離散コサイン変換若しくはカルーネン・レーベ変換等の直交変換と動き補償とにより画像圧縮を実現する画像情報符号化装置の概略構成を図８に示す。図８に示すように、画像情報符号化装置１００は、Ａ／Ｄ変換部１０１と、画面並べ替えバッファ１０２と、加算器１０３と、直交変換部１０４と、量子化部１０５と、可逆符号化部１０６と、蓄積バッファ１０７と、逆量子化部１０８と、逆直交変換部１０９と、フレームメモリ１１０と、動き予測・補償部１１１と、レート制御部１１２とにより構成されている。

【０００７】

図８において、Ａ／Ｄ変換部１０１は、入力された画像信号をデジタル信号に変換する。そして、画面並べ替えバッファ１０２は、Ａ／Ｄ変換部１０１から供給された画像圧縮情報のＧＯＰ（Group of Pictures）構造に応じて、フレームの並べ替えを行う。ここで、画面並び替えバッファ１０２は、イントラ（画像内）符号化が行われる画像に関しては、フレーム全体の画像情報を直交変換部１０４に供給する。直交変換部１０４は、画像情報に対して離散コサイン変換若しくはカルーネン・レーベ変換等の直交変換を施し、変換係数を量子化部１０５に供給する。量子化部１０５は、直交変換部１０４から供給された変換係数に対して量子化処理を施す。

【０００８】

可逆符号化部１０６は、量子化部１０５から供給された量子化された変換係数や量子化スケール等から符号化モードを決定し、この符号化モードに対して可変長符号化、又は算術符号化等の可逆符号化を施し、画像符号化単位のヘッダ部に挿入される情報を形成する

。符号化された符号化モードを蓄積バッファ１０７に供給して蓄積させる。この符号化された符号化モードは、画像圧縮情報として出力される。

【０００９】

また、可逆符号化部１０６は、量子化された変換係数に対して可変長符号化、若しくは算術符号化等の可逆符号化を施し、符号化された変換係数を蓄積バッファ１０７に供給して蓄積させる。この符号化された変換係数は、画像圧縮情報として出力される。

【００１０】

量子化部１０５の挙動は、レート制御部１１２によって制御される。また、量子化部１０５は、量子化後の変換係数を逆量子化部１０８に供給し、逆量子化部１０８は、その変換係数を逆量子化する。逆直交変換部１０９は、逆量子化された変換係数に対して逆直交変換処理を施して復号画像情報を生成し、その情報をフレームメモリ１１０に供給して蓄積させる。

【００１１】

一方、画面並び替えバッファ１０２は、インター（画像間）符号化が行われる画像に関しては、画像情報を動き予測・補償部１１１に供給する。動き予測・補償部１１１は、同時に参照される画像情報をフレームメモリ１１０より取り出し、動き予測・補償処理を施して参照画像情報を生成する。動き予測・補償部１１１は、この参照画像情報を加算器１０３に供給し、加算器１０３は、参照画像情報を当該画像情報との差分信号に変換する。また、動き補償・予測部１１１は、同時に動きベクトル情報を可逆符号化部１０６に供給する。

【００１２】

可逆符号化部１０６は、量子化部１０５から供給された量子化された変換係数や量子化スケールや、動き補償・予測部１１１から供給された動きベクトル情報等から符号化モードを決定し、この符号化モードに対して可変長符号化、又は算術符号化等の可逆符号化を施し、画像符号化単位のヘッダ部に挿入される情報を形成する。符号化された符号化モードを蓄積バッファ１０７に供給して蓄積させる。この符号化された符号化モードは、画像圧縮情報として出力される

【００１３】

また、可逆符号化部１０６は、その動きベクトル情報に対して可変長符号化若しくは算術符号化等の可逆符号化処理を施し、画像符号化単位のヘッダ部に挿入される情報を形成する。

【００１４】

イントラ符号化と異なり、インター符号化の場合には、直行変換部１０４に入力される画像情報は、加算器１０３より得られた差分信号である。

【００１５】

なお、その他の処理については、イントラ符号化を施される画像圧縮情報と同様であるため、説明を省略する。

【００１６】

続いて、上述した画像情報符号化装置１００に対応する画像情報復号装置の概略構成を図９に示す。図９に示すように、画像情報復号装置１２０は、蓄積バッファ１２１と、可逆復号部１２２と、逆量子化部１２３と、逆直交変換部１２４と、加算器１２５と、画面並べ替えバッファ１２６と、Ｄ／Ａ変換部１２７と、動き予測・補償部１２８と、フレームメモリ１２９とにより構成されている。

【００１７】

図９において、蓄積バッファ１２１は、入力された画像圧縮情報を一時的に格納した後、可逆復号部１２２に転送する。可逆復号部１２２は、定められた画像圧縮情報のフォーマットに基づき、画像圧縮情報に対して可変長復号若しくは算術復号等の処理を施し、ヘッダ部に格納された符号化モード情報を取得し逆量子化部１２３等に供給する。同様に、量子化された変換係数を取得し逆量子化部１２３に供給する。また、可逆復号部１２２は、当該フレームがインター符号化されたものである場合には、画像圧縮情報のヘッダ部に

格納された動きベクトル情報についても復号し、その情報を動き予測・補償部１２８に供給する。

【００１８】

逆量子化部１２３は、可逆復号部１２２から供給された量子化後の変換係数を逆量子化し、変換係数を逆直交変換部１２４に供給する。逆直交変換部１２４は、定められた画像圧縮情報のフォーマットに基づき、変換係数に対して逆離散コサイン変換若しくは逆カルーネン・レーベ変換等の逆直交変換を施す。

【００１９】

ここで、当該フレームがイントラ符号化されたものである場合には、逆直交変換処理が施された画像情報は、画面並べ替えバッファ１２６に格納され、Ｄ／Ａ変換部１２７におけるＤ／Ａ変換処理の後に出力される。

【００２０】

一方、当該フレームがインター符号化されたものである場合には、動き予測・補償部１２８は、可逆復号処理が施された動きベクトル情報とフレームメモリ１２９に格納された画像情報とに基づいて参照画像を生成し、加算器１２５に供給する。加算器１２５は、この参照画像と逆直交変換部１２４の出力とを合成する。なお、その他の処理については、イントラ符号化されたフレームと同様であるため、説明を省略する。

【００２１】

ここで、ＪＶＴにおける、可逆符号化部１０６について詳細に説明する。ＪＶＴの可逆符号化部１０６では、量子化部１０５や動き予測・補償部１１１から入力された入力されたモード情報や動き情報、量子化された係数情報といったシンボルに対して、図１０に示す様に、ＣＡＢＡＣ(Context-based Adaptive Binary Arithmetic Coding)と呼ばれる算術符号化（以下、ＣＡＢＡＣ）、もしくはＣＡＶＬＣ（Context-based Adaptive Variable Length Coding）と呼ばれる可変長符号化（以下、ＣＡＶＬＣ）のどちらかの可逆符号化が適用され、画像圧縮情報（ビットストリーム）が出力される。どちらの可逆符号化が適用されるかは図１０におけるＣＡＢＡＣ／ＣＡＶＬＣ選択情報により決められるものであり、このＣＡＢＡＣ／ＣＡＶＬＣ選択情報は画像情報符号化装置１００で決められ、ヘッダ情報としてビットストリームに埋め込まれて出力される。

【００２２】

まず、図１１に可逆符号化部１０６におけるＣＡＢＡＣの構成図を示す。図１１では、量子化部１０５や動き予測・補償部１１１から入力されたモード情報や動き情報、量子化された変換係数情報が多値シンボルとしてbinarization器１３１に入力される。binarization器１３１では、入力された多値シンボルを、あらかじめ決められた一定規則にもとづき任意の長さの２値シンボルの列に変換する。この２値シンボル列はＣＡＢＡＣ符号化器１３３に入力され、ＣＡＢＡＣ符号化器１３３では、入力された２値シンボルに対してバイナリシンボル算術符号化が適用され、その結果をビットストリームとして出力し、蓄積バッファ１０７に入力する。なお、Context演算器１３２では、binarization器１３１に入力されたシンボル情報とbinarization器１３１からの出力である２値シンボルをもとにContextの計算を行い、ＣＡＢＡＣ符号化器１３３に入力する。Context演算器１３２におけるContextメモリ群１３５には、符号化処理中に随時更新されるContextとリセット時などに用いられるContextの初期状態が保存される。

【００２３】

次に、図１２に可逆符号化部１０６におけるＣＡＶＬＣの構成図を示す。図１２では、量子化部１０５や動き予測・補償部１１１から入力されたモード情報や動き情報、量子化された変換係数情報が多値シンボルとしてとしてＣＡＶＬＣ符号化器１４０に入力される。ＣＡＶＬＣ符号化器１４０では、従来のＭＰＥＧなどで採用されている可変長符号化のように、入力された多値シンボルに対して可変長符号テーブルを適用して、ビットストリームを出力する。ここでContext保存器１４１では、既にＣＡＶＬＣ符号化器１４０で符号化された情報、例えば、処理中のブロックだけでなく既に処理されたブロックにおける各ブロック内の非０係数の個数や直前に符号化された係数の値などが保存される。ＣＡＶ

ＬＣ符号化器１４０は、このContext保存器１４１からの情報をもとにシンボルに適用する可変長符号テーブルを切り替えることが可能である。なお、Context保存器１４１にはリセット時などに用いられるContextの初期状態も保存される。この出力されたビットストリームは、蓄積バッファ１０７に入力される。

【００２４】

同様に、ＪＶＴにおける、可逆復号化部１２２について詳細に説明する。ＪＶＴの可逆復号化部１２２では、可逆符号化部１０６と同様に、入力されたビットストリームに対して、図１３に示す様に、ＣＡＢＡＣ、もしくはＣＡＶＬＣのどちらかの可逆復号化が適用される。どちらの可逆復号化が適用されるかは、入力されたビットストリームのヘッダ情報に埋め込まれたＣＡＢＡＣ／ＣＡＶＬＣ選択情報を読み込むことにより、ＣＡＢＡＣかＣＡＶＬＣのどちらかを適用する。

【００２５】

図１４に可逆復号化部１２２におけるＣＡＢＡＣの構成図を示す。図１４では、蓄積バッファ１２１より入力されたビットストリームに対しＣＡＢＡＣ復号化器１６１においてバイナリシンボル算術復号化が適用され、その結果が２値シンボル列として出力される。この２値シンボル列は、逆binarization器１６３に入力され、逆binarization器１６３において、あらかじめ決められた一定規則にもとづき多値シンボルに変換される。この逆binarization器１６３から出力される多値シンボルは、モード情報や動きベクトル、量子化された係数情報として、逆binarization器１６３から出力され、逆量子化部１２３、動き予測・補償部１２８に入力される。なお、Context演算器１６２では、逆binarization器１６３に入力された２値化シンボル列と逆binarization器１６３からの出力である多値シンボルをもとにContextの計算を行い、ＣＡＢＡＣ復号化器１６１に入力する。Context演算器１６２におけるContextメモリ群１６５には、復号化処理中に随時更新されるContextとリセット時などに用いられるContextの初期状態が保存される。

【００２６】

次に、図１５に可逆復号化部１２２におけるＣＡＶＬＣの構成図を示す。図１５では、蓄積バッファ１２１より入力されたビットストリームがＣＡＶＬＣ復号化器１７０に入力される。ＣＡＶＬＣ復号化器１７０では、従来のＭＰＥＧなどで採用されている可変長復号化のように、入力されたビットストリームに対して可変長復号テーブルを適用して、モード情報や動き情報、量子化された変換係数情報を出力する。これら出力情報は、可逆量子化部１２３、動き予測・補償部１２８に入力される。ここでContext保存器１７１では、既にＣＡＶＬＣ復号化器１７０で復号化された情報、例えば、処理中のブロックだけでなく既に処理されたブロックにおける各ブロック内の非０係数の個数や直前に復号化された係数の値などが保存される。ＣＡＶＬＣ復号化器１７０は、このContext保存器１１からの情報をもとにシンボルに適用する可変長復号テーブルを切り替えることが可能である。なお、Context保存器１４１にはリセット時などに用いられるContextの初期状態も保存される。

【００２７】

この図１１、図１４に示すＣＡＢＡＣの詳細動作として、以下にＦｉｎａｌ　Ｃｏｍｍｉｔｔｅｅ　Ｄｒａｆｔ　ＩＳＯ／ＩＥＣ　14496-10：2002（第９．２節）におけるＣＡＢＡＣの説明を添付する（例えば、非特許文献１参照。）。

【００２８】

9.2Context-based adaptive binary arithmetic coding (CABAC)

9.2.1 Decoding flow and binarization

Binarizationとはnon-binary symbolからbinary列(binと呼ばれる)への変換を行う処理のことであり、9.2.1.1 - 9.2.1.4節において、CABACの為の基本的なbinarization方式が規定される。Decoding flow、及び、全てのsyntax elementに対するbinarization方法は9.2.1.5 - 9.2.1.9 節で規定される。

【００２９】

9.2.1.1 Unary binarization

Unary codeによるbinalizationの最初の5 symbolに対する表をTable 9-19に示す。

【００３０】

Code symbol C に対しては、C個の ’１’ の最後に ’０’ を付けたbinary列が対応する。Binの最初のbitにはbin number =1が対応し、2番目のbitにはbin number=2、と、最後のbitに行くに従って対応するbin numberは増えていく。

【００３１】

【表１】

Table 9-19 – Binarization by means of the unary code tree

Code symbol	Binarization						
0	0						
1	1	0					
2	1	1	0				
3	1	1	1	0			
4	1	1	1	1	0		
5	1	1	1	1	1	0	
bin_num	1	2	3	4	5	6	

【００３２】

9.2.1.2 Truncated unary (TU) binarization

Truncated unary (TU) binarizationは有限個のシンボル{0,…,C_{max}}に対して適用される。Symbol C＜C_{max};に対しては9.2.1.1節で規定されたunnary binarizationを行い、シンボルC_{max}にはC_{max} 個の1を割り当てる。Bin numberの割り振り方はunary binarizationの場合と同じである。

【００３３】

9.2.1.3 Concatenated unary/ k^{th}-order Exp-Golomb (UEGk) binarization

Concatenated unary/k^{th}-order Exp-Golomb (UEGk) binarization は、C_{max}=Ucoff (Ucoff:Cut off parameter)としたtruncated unary binarization code (prefix code)とk次のExp-Golomb符号 (suffix code)とが連接されたものが変換後のbinary列となる。Symbol C がC＜Ucoffの場合suffix codeは無く、C≧Ucoffの場合suffix codeはsymbol C-Ucoff に対するExp-Golomb符号となる。

【００３４】

Symbol S に対するk次のExp-Golomb codeは以下のように構成される:

```
while(1) {
      //first unary part of EGk
      if (symbol>= (unsigned int)(1<<k)){
            put(‘1’);
            S = S - (1<<k);
            k++;’
      }
      else
        {
            put(‘0’);    //now terminating zero of unary part of EGk
            while (k--)  //finally binary part of EGk
                  put( (S>>k)&0x01 );
            break;
        }
}
```

【００３５】

Bin numberは、unary codeの第1ビット目をbin_num=1として、Exp-Golomb符号のLSBに

向かって1づつ増えていく。

【００３６】

9.2.1.4 Fixed-length (FL) binarization

有限個のシンボル{0 , …, C_{max}} に対し、L-bit (L= $\log_2 | C_{max} | +1$)のbinarizationを適用する。Bin numberはLSBをbin_num=1とし、MSBに向かって増えていく。

【００３７】

9.2.1.5 Binarization schemes for macroblock type and sub macroblock typeI slice中のmacroblock typeのbinarization方式はTable 9-20で規定される。ただし、adaptive_block_size_transform_flag==1であった場合には、Table 12-10に従う。

【００３８】

SI slice中のmacroblock typeのbinarization後のbit列はprefixとsuffix部分からなり、prefixはb_1= ((mb_type = = SIntra_4x4) ? 0 : 1) で表される1bit、suffixはSintra4x4の場合(suffix無し)を除きTable 9-20に示すbinarization patternに基づく。

【００３９】

P, SP, B sliceのbinarizationはTable 9-21で規定される。P, SP slice中のintra macroblock type (mb_type値7～30に相当)は、Table 9-21に示すprefixとTable 9-20に示すsuffixによってbinarizationが行われる。

【００４０】

B slice中のintra macroblock type (mb_type値23～47に相当)についても、Table 9-21に示すprefixとTable 9-20に示すsuffixによってbinarizationが行われる。adaptive_block_size_transform_flag==1の場合は、Table 9-21における対応するsliceのprefixと、Table 12-10で規定されるsuffixが用いられる。

【００４１】

P, SP, B sliceにおけるsub_mb_typeのbinarizationはTable 9-22で与えられる。

【００４２】

【表２】

Table 9-20 – Binarization for macroblock types for I slices

Value (name) of mb_type	Binarization					
0 (Intra_4x4)	0					
1 (Intra_16x16_0_0_0)	1	0	0	0	0	
2 (Intra_16x16_1_0_0)	1	0	0	0	1	
3 (Intra_16x16_2_0_0)	1	0	0	1	0	
4 (Intra_16x16_3_0_0)	1	0	0	1	1	
5 (Intra_16x16_0_1_0)	1	0	1	0	0	0
6 (Intra_16x16_1_1_0)	1	0	1	0	0	1
7 (Intra_16x16_2_1_0)	1	0	1	0	1	0
8 (Intra_16x16_3_1_0)	1	0	1	0	1	1
9 (Intra_16x16_0_2_0)	1	0	1	1	0	0
10 (Intra_16x16_1_2_0)	1	0	1	1	0	1
11 (Intra_16x16_2_2_0)	1	0	1	1	1	0
12 (Intra_16x16_3_2_0)	1	0	1	1	1	1
13 (Intra_16x16_0_0_1)	1	1	0	0	0	
14 (Intra_16x16_1_0_1)	1	1	0	0	1	
15 (Intra_16x16_2_0_1)	1	1	0	1	0	
16 (Intra_16x16_3_0_1)	1	1	0	1	1	
17 (Intra_16x16_0_1_1)	1	1	1	0	0	0
18 (Intra_16x16_1_1_1)	1	1	1	0	0	1
19 (Intra_16x16_2_1_1)	1	1	1	0	1	0
20 (Intra_16x16_3_1_1)	1	1	1	0	1	1
21 (Intra_16x16_0_2_1)	1	1	1	1	0	0
22 (Intra_16x16_1_2_1)	1	1	1	1	0	1
23 (Intra_16x16_2_2_1)	1	1	1	1	1	0
24 (Intra_16x16_3_2_1)	1	1	1	1	1	1
bin_num	1	2	3	4	5	6

【００４３】

【表３】

Table 9-21 – Binarization for macroblock types for P, SP, and B slices

Slice type	Value (name) of mb_type	Binarization						
P, SP slices	0 (Pred_L0_16x16)	0	0	0				
	1 (Pred_L0_L0_16x8)	0	1	1				
	2 (Pred_L0_L0_8x16)	0	1	0				
	4 (Pred_8x8)	0	0	1				
	6 (Pred_8x8ref0)	na						
	7 to 30 (Intra, prefix only)	1						
B slices	0 (Direct_16x16)	0						
	1 (Pred_L0_16x16)	1	0	0				
	2 (BiPred_L1_16x16)	1	0	1				
	3 (BiPred_Bi_16x16)	1	1	0	0	0	0	
	4 (Pred_L0_L0_16x8)	1	1	0	0	0	1	
	5 (Pred_L0_L0_8x16)	1	1	0	0	1	0	
	6 (BiPred_L1_L1_16x8)	1	1	0	0	1	1	
	7 (BiPred_L1_L1_8x16)	1	1	0	1	0	0	
	8 (BiPred_L0_L1_16x8)	1	1	0	1	0	1	
	9 (BiPred_L0_L1_8x16)	1	1	0	1	1	0	
	10 (BiPred_L1_L0_16x8)	1	1	0	1	1	1	
	11 (BiPred_L1_L0_8x16)	1	1	1	1	1	0	
	12 (BiPred_L0_Bi_16x8)	1	1	1	0	0	0	0
	13 (BiPred_L0_Bi_8x16)	1	1	1	0	0	0	1
	14 (BiPred_L1_Bi_16x8)	1	1	1	0	0	1	0
	15 (BiPred_L1_Bi_8x16)	1	1	1	0	0	1	1
	16 (BiPred_Bi_L0_16x8)	1	1	1	0	1	0	0
	17 (BiPred_Bi_L0_8x16)	1	1	1	0	1	0	1
	18 (BiPred_Bi_L1_16x8)	1	1	1	0	1	1	0
	19 (BiPred_Bi_L1_8x16)	1	1	1	0	1	1	1
	20 (BiPred_Bi_Bi_16x8)	1	1	1	1	0	0	0
	21 (BiPred_Bi_Bi_8x16)	1	1	1	1	0	0	1
	22 (BiPred_8x8)	1	1	1	1	1	1	
	23 to 47 (Intra, prefix only)	1	1	1	1	0	1	
bin_num		1	2	3	4	5	6	7

【００４４】

【表４】

Table 9-22 – Binarization for sub macroblock types in P and B slices

Slice type	Value (name) of sub_mb_type	Binarization					
P slices	0 (Pred_L0_8x8)	1					
	1 (Pred_L0_8x4)	0	0	0			
	2 (Pred_L0_4x8)	0	0	1	1		
	3 (Pred_L0_4x4)	0	0	1	0		
	4 (Intra_8x8)	0	1				
B slices	0 (Direct_8x8)	0					
	1 (Pred_L0_8x8)	1	0	0			
	2 (BiPred_L1_8x8)	1	0	1			
	3 (BiPred_Bi_8x8)	1	1	0	0	0	
	4 (Pred_L0_8x4)	1	1	0	0	1	
	5 (Pred_L0_4x8)	1	1	0	1	0	
	6 (BiPred_L1_8x4)	1	1	0	1	1	
	7 (BiPred_L1_4x8)	1	1	1	0	0	0
	8 (BiPred_Bi_8x4)	1	1	1	0	0	1
	9 (BiPred_Bi_4x8)	1	1	1	0	1	0
	10 (Pred_L0_4x4)	1	1	1	0	1	1
	11 (BiPred_L1_4x4)	1	1	1	1	0	0
	12 (BiPred_Bi_4x4)	1	1	1	1	0	1
	13 (Intra_8x8)	1	1	1	1	1	
bin_num		1	2	3	4	5	6

【００４５】

9.2.1.6 Decoding flow and assignment of binarization schemes

この節においてcoded_block_pattern, delta_qp 及びreference picture index, motion vector data, Intra4x4 prediction modesそれぞれのsyntax elementのbinarization方式を規定する。

【００４６】

基本的にcoded block patternは7.4.6節で規定された関係coded_block_pattern = coded_block_patternY + 16*ncによって復号される。最初にcoded_block_pattern 中のcoded_block_patternYが、C_{max} = 15 , L = 4によるfixed-length (FL) binarizationによって復号され、次に、色差のncがC_{max} = 2のTU binarizationによって復号される。

【００４７】

delta_qp parameterの復号は次に示すように2段階で行われる。最初にunsigned 値wrapped_delta_qp≧0がunary binarizationによって復号され、次に、Table 9-2に示される対応関係によって符号付きの値に直される。

【００４８】

Intra_4x4, Sintra_4x4のlumaに対するspatial intra prediction modesの復号は次のように規定される。最初にintra_pred_indicatorがC_{max} = 8のtruncated unary (TU) binarizationによって復号される。もし、intra_pred_indicatorが0であった場合、use_most_probable_mode = 1とし、intra_pred_indicator≧1の場合には、remaining_mode_selector = intra_pred_indicator - 1とする。intra_pred_modeは与えられたmost_probable_mode、remaining_mode_selectorを用い、9.1.5.1節で規定される方法で復号が行われる。復号順序はFigure 9-1 b)に示されるものと同様である。Chromaのintra_chroma_pred_modeに対しては、C_{max}=3のtruncated unary (TU) binarizationを用いて復号が行われる。

【００４９】

Reference picture index parameterは9.2.1.1で規定されるunary binarizationを用いて復号される。

【００５０】

動きベクトルの符号化された各コンポーネントは、それぞれのコンポーネント毎に復号される。それぞれのコンポーネントは水平、垂直成分を含むが、水平方向に対応するものが最初に復号される。最初に絶対値abs_mvd_comp、次に、符号sign_mvd_compが復号される。abs_mvd_compに適用されるbinarizationはcut-off parameter Ucoff = 9のconcatenated unary/3rd-order Exp-Golomb (UEG3) binarizationである。Exp-Golomb復号の際には、9.2.4.3.5で規定されているDecode_eq_prob処理が適用される。

【００５１】

9.2.1.7 Decoding flow and binarization of transform coefficients変換係数の復号は3段階からなる。Macroblock levelでのcoded_block_patternによって係数値があることが分かっている場合、それぞれのblockに対するcoded_block_flagが復号されるが、coded_block_flagが0である場合、当該blockに対する以降の情報は復号されない。coded_block_flag != 0の場合、scanの最後の位置を除くそれぞれのscan位置 iに対するsignificant_coeff_flag[i]を復号する。significant_coeff_flag[i]が1であった場合、次にlast_significant_coeff_flag[i]が復号される。last_significant_coeff_flag[i]が1であるということは、scan位置iの係数値がscanパス順で現れる最後の係数であることを意味する。last_significant_coeff_flag[i]が1となった時には、次に、coeff_absolute_value_minus_1をscanの逆順で復号し、同様に、その次coeff_signをscanの逆順で復号する。coeff_absolute_value_minus_1はUCoff=14のunary/zero-order Exp-Golomb (UEG0) binarizationを用いて復号される。動きベクトルの絶対値復号の場合と同様に、Exp-Golomb suffixはDecode_eq_prob処理を用いて復号される。

【００５２】

9.2.1.8 Decoding of sign information related to motion vector data and transform coefficients

動きベクトルの符号情報sign_mvd_compと係数値の符号情報coeff_signは以下のように復号される。最初に9.2.4.3.5節で規定されるDecode_eq_prob処理を行って得られたsign_indを用い、符号情報sign_infoをsign_info = ((sign_ind = = 0) ? 1 : -1)によって得る。

【００５３】

9.2.1.9 Decoding of macroblock skip flag and end-of-slice flag

mb_skip_flagの復号は以下のように行われる。最初に9.2.2.2節で規定されたcontext modelを用いmb_skip_flag_decodedを復号する。次にmb_skip_flagをmb_skip_flag_decodedを反転する(i.e., mb_skip_flag = mb_skip_flag_decoded ^ 0x01)ことによって得る。

【００５４】

end_of_slice_flagはState = 63, MPS = 0のfixed, non-adaptive modelによって復号される。この場合、以下に示す理由によって、9.2.4.2に示される確率予測を各復号ステップで行っているにも関わらず、fixed modeとなることが示される。end_of_slice valuesがずっと’0’であった場合、 MPS symbolの観測によってもState=63は確率予測の結果State=63のままとなる。そして、LPS値’1’がend_of_slice_flagの値として復号された場合は、この時点でsliceの終わりに到達していることになるから以降の復号処理には影響しない。以上のような理由でState = 63, MPS = 0に設定することにより、fixed, non-adaptive modelが実現される。

【００５５】

9.2.2 Context definition and assignment

それぞれのbin numberに対して、それまでに復号されたsymbol等を含む諸条件によって決まるcontext variableが定義される。Context variableの値は特定のbin numberに対するcontext modelを定める。それぞれのbin number : bin_num に対して複数のcontext labelが定めらる場合もあるが、1つだけの場合もある。

【００５６】

この節ではsyntax elementを符号化する為の一般的なcontext variableの算出法であるcontext templatesを定義し、syntax elementのそれぞれのbin numberに対応するcontext

variableを規定する。まず、それぞのれsyntax elementの異なるbin numberに対して与えられるcontext variableを規定する為にcontext identifier: context_idを規定するが、これは、bin number k に対するcontext variableをcontext_id[k]として表すためのものである。このcontext_id[k]は、1≦k≦N (N=max_idx_ctx_id)の範囲で規定される。

【００５７】

Table 9-23にそれぞれのsyntax elementのカテゴリ毎のcontext identifierの概要を示す。

【００５８】

対応するcontext variableについてのより詳細な記述は以降の節において記述する。それぞれのcontext identifierはcontext labelの特定の範囲に対応する。macroblock typeの場合はI, SI, P, SP,B.のそれぞれについて別々のcontext identifierが存在しており、個別のcontext labelの範囲を持つが、context labelの範囲自体は重複している。

【００５９】

変換係数のcontext identifierの場合、adaptive_block_size_transform_flag==1の時にはTable 12-12に示される追加のcontext label値を用いる。

【００６０】

【表５】

Table 9-23 – Syntax elements and associated context identifiers

Syntax element	Context identifier	Type of Binarization	max_idx_ctx_id	Range of context label
mb_skip_flag	ctx_mb_skip	-/-	1	0 – 2
mb_type	ctx_mb_type_I	Table 9-14	5	0 – 7
	ctx_mb_type_SI_pref	-/-	1	0 – 2
	ctx_mb_type_SI_suf	Table 9-14	5	3 – 10
	ctx_mb_type_P	Table 9-15	3	3 – 6
	ctx_mb_type_B		4	3 – 8
	ctx_mb_type_P_suf	Table 9-14	5	6 – 9
	ctx_mb_type_B_suf		5	8 – 11
	ctx_b8_mode_P	Table 9-16	4	12 – 15
	ctx_b8_mode_B		4	12 – 15
mvd_l0 and mvd_l1	ctx_abs_mvd_h	UEG3, UCoff=9	5	16 – 22
	ctx_abs_mvd_v	UEG3, UCoff=9	5	23 – 29
ref_idx_l0 and ref_idx_l1	ctx_ref_idx	Unary	3	30 – 35
delta_qp	ctx_delta_qp	Unary	3	36 – 39
chroma_pred_mode	ctx_ipred_chroma	TU, C_{max}=3	3	40 – 44
intra_pred_mode	ctx_ipred_luma	TU, C_{max}=8	2	45 – 62
coded_block_pattern	ctx_cbp_luma	FL, C_{max}=15	4	63 – 66
	ctx_cbp_chroma	TU, C_{max}=2	2	67 – 74
coded_block_flag	ctx_cbp4	-/-	-/-	75 – 94
significant_coeff	ctx_sig	-/-	-/-	95 – 155
last_coeff	ctx_last	-/-	-/-	156 – 216
coeff_absolute_value_minus_1	ctx_abs_level	UEG0, UCoff=14	2	217 – 266
end_of_slice_flag	ctx_eos	Fixed, non-adaptive model with State(ctx_eos) = 63, MPS(ctx_eos) = 0		

【００６１】

9.2.2.1 Overview of assignment of context labels

Table 9-24, 9-25にcontext identifiersとそのcontext labelの範囲を示す。このcontext label(実際にはoffsetが加算されたものが参照label番号となる)とbin numberの関係によって、どのcontext variableがfixed modelを用い、どのcontext variableが複数のmodelを持つのかが分かる。

【００６２】

Table 9-24の特定のbin number bin_numに複数のcontext labelが割り振られているもの、また、Table 9-25におけるcontext_categoryに対して複数のcontext labelが与えられているものは、複数のmodelからの選択を行う。

【００６３】

【表６】

Table 9-24 – Overview of context identifiers and associated context labels

Context identifier	Range of context label	Offset for context label	max_idx_ctx_id	bin_num				
				1	2	3	4	≥ 5
ctx_mb_skip	0 – 2	0	1	0, 1, 2	-/-	-/-	-/-	-/-
ctx_mb_type_I	0 – 7	0	5	0, 1, 2	3	4	5, 6	6, 7
ctx_mb_type_SI_pref	0 – 2	0	1	0, 1, 2	-/-	-/-	-/-	-/-
ctx_mb_type_SI_suf	3 – 10	3	5	0, 1, 2	3	4	5, 6	6, 7
ctx_mb_type_P	3 – 6	3	3	0	1	2, 3	-/-	-/-
ctx_mb_type_P_suf	6 – 9	6	5	0	1	2	2, 3	3
ctx_mb_type_B	3 – 8	3	4	0, 1, 2	3	4, 5	5	5
ctx_mb_type_B_suf	8 – 11	8	5	0	1	2	2, 3	3
ctx_b8_mode_P	12 – 15	12	4	0	1	2	3	-/-
ctx_b8_mode_B	12 – 15	12	4	0	1	2, 3	3	3
ctx_abs_mvd_h	16 – 22	16	5	0, 1, 2	3	4	5	6
ctx_abs_mvd_v	23 – 29	23	5	0, 1, 2	3	4	5	6
ctx_ref_idx	30 – 35	30	3	0, 1, 2, 3	4	5	5	5
ctx_delta_qp	36 – 39	36	3	0, 1	2	3	3	3
ctx_ipred_chroma	40 – 44	36	3	0, 1, 2	3	4	-/-	-/-
ctx_ipred_luma	45 – 62	42	2	0, . . ., 8	9, . . ., 17	9, . . ., 17	9, . . ., 17	9, . . ., 17
ctx_cbp_luma	63 – 66	60	4	0, 1, 2, 3	0, 1, 2, 3	0, 1, 2, 3	0, 1, 2, 3	-/-
ctx_cbp_chroma	67 – 74	64	2	0, 1, 2, 3	4, 5, 6, 7	-/-	-/-	-/-
ctx_abs_level	217 – 266	217+ 10*context_category	2	0, . . ., 4	5, . . ., 9	5, . . ., 9	5, . . ., 9	5, . . ., 9

【００６４】

【表７】

Table 9-25 – Overview of context identifiers and associated context labels

(continued)

Context identifier	Offset (range) of context label	context_category (as specified in Table 9-22)				
		0	1	2	3	4
ctx_cbp4	75 (75 – 94)	0 – 3	4 – 7	8 – 11	12 – 15	16 – 19
ctx_sig	95 (95 – 155)	0 – 14	15 – 28	29 – 43	44 –46	47 – 60
ctx_last	156 (156 – 216)	0 – 14	15 – 28	29 – 43	44 –46	47 – 60

【００６５】

9.2.2.2 Context templates using two neighbouring symbols

一般的なcontext variableの設定方法を説明するためにFigure 9-2（図２２）を用いる。当該block Cに対して隣接する左blockと上blockにおける同一syntax elementのsymbolまたはbinがA,Bとして図示されている。

【００６６】

Contextを決める式の第一番目は以下の通りとなる。

$$ctx\_var\_spat = cond\_term(A, B), \quad (9\text{-}1)$$

cond_term(A, B)は隣接symbol A,Bとcontext variableの間の関係を表す関数である。

【００６７】

この他に、3つのテンプレートが以下のように定義される。

$$ctx\_var\_spat1 = cond\_term(A) + cond\_term(B), \quad (9\text{-}2)$$

$$ctx\_var\_spat2 = cond\_term(A) + 2*cond\_term(B), \quad (9\text{-}3)$$

$$ctx\_var\_spat3 = cond\_term(A). \quad (9\text{-}4)$$

Table 9-26 において2つの隣接symbolからのcontext variableの求め方を示す。

ctx_cbp4はTable 9-28に示される6つのblock type(Luma-DC, Luma-AC, Chroma-U-DC, Chroma-V-DC, Chroma-U-AC, Chroma-V-AC)によって決定される。

【００６８】

【表８】

Table 9-26 – Specification of context variables using context templates according to Equations (9-2) – (9-4)

Context variable	Context template	cond_term(X), semantics of X	cond_term(X), if X not available
ctx_mb_skip	*ctx_var_spat1*	(mb_skip_flag(X) == 0) ? 1: 0 X: neighbouring macroblock	0
ctx_mb_type_I[1]	*ctx_var_spat1*	(mb_type(X) != Intra_4x4) ? 1: 0 X: neighbouring macroblock	0
ctx_mb_type_SI_pref[1]	*ctx_var_spat1*	(mb_type(X) != Sintra_4x4) ? 1: 0 X: neighbouring macroblock	0
ctx_mb_type_SI_suf[1]	*ctx_var_spat1*	(mb_type(X) != Intra_4x4) ? 1: 0 X: neighbouring macroblock	0
ctx_mb_type_B[1]	*ctx_var_spat1*	((mb_type(X) != Direct) ? 1: 0) X: neighbouring macroblock	0
ctx_ipred_chroma[1]	*ctx_var_spat1*	((intra_chroma_pred_mode(X) != 0) ? 1: 0) X: neighbouring macroblock	0
ctx_ref_idx[1]	*ctx_var_spat2*	(ref_idx_l0/ref_idx_l1(X) != 0) ? 1: 0 X: neighbouring block	0
ctx_ipred_luma[i], i=1, 2	*ctx_var_spat3*	9*(i-1) + intra_pred_mode(X) X: neighbouring block	0
ctx_cbp_luma[i], i=1, ..., 4	*ctx_var_spat2*	i-th bit of coded_block_patternY(X) X: neighbouring 8x8 block of i-th block	0
ctx_cbp_chroma[1]	*ctx_var_spat2*	(nc(X) != 0) ? 1: 0 X: neighbouring macroblock	0
ctx_cbp_chroma[2]	*ctx_var_spat2*	4 + (nc(X) == 2) ? 1: 0 X: neighbouring macroblock	0
ctx_cbp4	*ctx_var_spat2*	coded_block_pattern_4(X) X: neighbouring block of same block type	1, if X is INTRA; 0, otherwise
ctx_delta_qp	*ctx_var_spat3*	(delta_qp(X) != 0) ? 1: 0 X: neighbouring macroblock	0

【００６９】

【数１】

$$ctx\_abs\_mvd\_comp[1]=\begin{cases}0, (|mvd\_comp(A)|+|mvd\_comp(B)|)<3,\\ 2, (|mvd\_comp(A)|+|mvd\_comp(B)|)>32,\\ 1, otherwise,\end{cases} \quad (9-5)$$

【００７０】

compは水平成分(h)または垂直性分(v)を意味し、A，BはFigure 9-2に示すような隣接blockを意味する。これら隣接blockは異なるmacroblock partitionに属する可能性がある為、以下のような隣接blockを特定する為の方法が規定されている。最初に4x4 blockの動きベクトルはoversampleされている、つまり、対応するblockがより粗くpartitioningされていた場合、quadtreeにおける親blockの動きベクトルを継承しているとみなす。逆に、当該block Cが隣接blockより粗くpartitioningされていた場合、隣接blockの左上のsub-blockの動きベクトルを対応動きベクトルとする。これらの処理によって隣接blockにおける対応する値を求めた後、(9-5)を用いてcontext variableを得る。

【００７１】

9.2.2.3　Context templates using preceding bin values

$(b_1, \cdots, b_N)$ がsymbol Cのbinarizationに相当すると仮定した場合、Cのk番目のbinに対応するcontext variableは以下のように規定される。

$$ctx\_var\_bin[k] = cond\_term(b_1, \cdots, b_{k-1}), \quad (9\text{-}6)$$

ただし、1＜k≦Nとする。Table 9-27において、この種のcontext variableの与え方の一覧を示す。

【００７２】

【表９】

Table 9-27 – Definition of context variables using the context template according to Equation (9-6)

Context variable	cond_term($b_1, \ldots, b_{k-1}$)
ctx_mb_type_I[4]	(b_3 = = 0) ? 5: 6
ctx_mb_type_I[5]	(b_3 = = 0) ? 6: 7
ctx_mb_type_SI_suf[4]	(b_3 = = 0) ? 5: 6
ctx_mb_type_SI_suf[5]	(b_3 = = 0) ? 6: 7
ctx_mb_type_P[3]	(b_2 = = 0) ? 2: 3
ctx_mb_type_P_suf[4]	(b_3 = = 0) ? 2: 3
ctx_mb_type_B[3]	(b_2 = = 1) ? 4: 5
ctx_mb_type_B_suf[4]	(b_3 = = 0) ? 2: 3
ctx_b8_mode_B[3]	(b_2 = = 1) ? 2: 3

9.2.2.4　Additional context definitions for information related to transform coefficients

変換係数の条件付けの為には、さらに3つの異なるcontext identifierが追加で用いられる。

【００７３】

これらのcontext identifierはTable 9-28で示されるcontext_categoryに依存して決まる。

【００７４】

【表１０】

Table 9-28 – Context categories for the different block types

Block_type	MaxNumCoeff	context_category
Luma DC block for INTRA16x16 mode	16	0:Luma-Intra16-DC
Luma AC block for INTRA16x16 mode	15	1:Luma-Intra16-AC
Luma block for INTRA 4x4 mode	16	2:Luma-4x4
Luma block for INTER 4x4 mode	16	
U-Chroma DC block for INTRA mode	4	3:Chroma-DC
V-Chroma DC block for INTRA mode	4	
U-Chroma DC block for INTER mode	4	
V-Chroma DC block for INTER mode	4	
U-Chroma AC block for INTRA mode	15	4:Chroma-AC
V-Chroma AC block for INTRA mode	15	
U-Chroma AC block for INTER mode	15	
V-Chroma AC block for INTER mode	15	

【００７５】

adaptive_block_size_transform_flag==1の場合には12.5.2節で規定されるようなcontext categoryがさらに加わる。Context identifier ctx_sigとctx_lastはbinary値を持つSIG とLAST、及び、当該blockのscanning_posによって以下のように与えられる。

【００７６】

ctx_sig[scanning_pos] = Map_sig(scanning_pos), (9-7)

ctx_last[scanning_pos] = Map_last(scanning_pos). (9-8)

(9-7) と (9-8) にあるMap_sigとMap_lastはblock typeに依存して変わる。

【００７７】

まず、context categoryが0-4の場合、それぞれ以下のような恒等写像となる。

【００７８】

Map_sig(scanning_pos)=Map_last(scanning_pos) = scanning_pos, if context_category =0,…,4,

scanning_pos はzig-zag scanにおけるscan位置を示す。adaptive_block_size_transform_flag==1の場合のみに使われるcontext category=5 ～7 に対するMap_sig, Map_lastは12.5.2節で与えられる。

【００７９】

変換係数-1を表すabs_level_m1の復号の際にはctx_abs_levelがcontext identifierとして用いられる。ctx_abs_levelはctx_abs_level[1] とctx_abs_level[2]の2つのcontext variableの値を用いて以下のように求められる。

ctx_abs_lev[1] = ((num_decod_abs_lev_gt1!=0) ? 4: min(3, num_decod_abs_lev_eq1)), (9-9)

ctx_abs_lev[2] = min(4, num_decod_abs_lev_gt1), (9-10)

num_decod_abs_lev_eq1は係数値が1の係数の数を表し、num_decod_abs_lev_gt1は係数値

が1より大きな係数の数を表す。Context variable ctx_abs_level[k], k=1,2 を算出する際には当該blockの変換係数のみが必要であり、他の情報は必要とされない。

【００８０】

9.2.3 Initialisation of context models

9.2.3.1 Initialisation procedure

Sliceの先頭でstate numberと9.2.4.2で定義されるMPSに対応するシンボルが初期化される。それら2つを合わせて初期状態と呼ぶが、実際の初期状態は量子化パラメータQPによって以下のように定められる。

【００８１】

pre_stateをpre_state = ((m*(QP-12))＞＞4) + nによって算出する。

pre_state を P, B sliceの場合[0,101] に、I slice の場合[27,74]にクリップする。処理は以下の通り。

【００８２】

pre_state = min (101, max(0,pre_state)) for P- and B-slices and
pre_state = min (74, max(27,pre_state)) for I-slices;
pre_state から {state, MPS} へのmappingを以下の式に従って行う:

if (pre_state <= 50) then {state = 50-pre_state, MPS = 0} else {state = pre_state-51, MPS = 1}

【００８３】

9.2.3.2 Initialisation procedure

Tables 9-29－9-34に全てのsyntax elementに対する初期化parameterを示す。初期状態は9.2.3.1節に述べた方法によって得られる。

【００８４】

【表１１】

Table 9-29 – Initialisation parameters for context identifiers *ctx_mb_type_I, ctx_mb_type_SI_pref, , ctx_mb_type_SI_suf, ctx_mb_skip, ctx_mb_type_P, ctx_mb_type_B*

Context label	ctx_mb_type_I		ctx_mb_type_SI_pref		ctx_mb_skip					
	m	n	m	n	m	n	m	n	m	n
0	7	25	7	25	-23	66				
1	8	35	8	35	-14	77				
2	-2	63	-2	63	-9	88				
			ctx_mb_type_SI_suf				ctx_mb_type_P		ctx_mb_type_B	
3	-9	68	7	25			2	13	9	49
4	-15	74	8	35			14	24	3	65
5	-3	36	-2	63			-21	69	0	78
6	-1	51	-9	68			-1	52	-13	81
7	0	50	-15	74					-14	73
8			-3	36					-8	64
9			-1	51						
10			0	50						

【００８５】

【表１２】

Table 9-30 – Initialisation parameters for context identifiers *ctx_b8_mode_P, ctx_b8_mode_B, ctx_mb_type_P_suf, ctx_mb_type_B_suf*

Context label	ctx_mb_type_P_suf		Context label	ctx_b8_mode_P		ctx_b8_mode_B	
	m	n		m	n	m	n
9	-9	55	12	8	46	-9	62
10	-7	50	13	12	11	-12	66
11	2	47	14	-4	62	-9	56
			15	18	48	3	47

【００８６】

【表１３】

Table 9-31 – Initialisation parameters for context identifiers *ctx_abs_mvd_h, ctx_abs_mvd_v, ctx_ref_idx*

Context label	ctx_abs_mvd_h		Context label	ctx_abs_mvd_v		Context label	ctx_ref_idx	
	m	n		m	n		m	n
16	1	48	23	-1	45	30	3	27
17	-5	60	24	-5	59	31	-1	47
18	-8	70	25	-9	71	32	0	45
19	2	52	26	0	50	33	-2	60
20	2	62	27	3	61	34	-1	57
21	-3	64	28	-3	63	35	0	48
22	-2	80	29	1	80			

【００８７】

【表１４】

Table 9-32 – Initialisation parameters for context identifiers *ctx_delta_qp, ctx_ipred_chroma, ctx_ipred_luma*

Context label	ctx_delta_qp		Context label	ctx_ipred_luma		Context label	ctx_ipred_luma	
	m	n		m	n		m	n
36	0	28	45	-5	49	54	-4	61
37	0	50	46	-3	58	55	-6	64
38	0	50	47	-5	58	56	-6	63
39	0	50	48	-4	58	57	-3	75
	ctx_ipred_chrom		49	-5	59	58	-4	63
40	-5	50	50	-4	60	59	-7	65
41	0	50	51	-6	61	60	-1	63
42	0	50	52	-5	62	61	-18	66
43	0	50	53	-4	60	62	-9	67
44	0	50						

【００８８】

【表１５】

Table 9-33 – Initialisation parameters for context identifiers *ctx_cbp_luma, ctx_cbp_chroma*

Context label	ctx_cbp_luma				Context label	ctx_cbp_chroma				Context label	ctx_cbp_chroma			
	I slices		P, B slices			I slices		P, B slices			I slices		P, B slices	
	m	n	m	n		m	n	m	n		m	n	m	n
63	-3	75	-21	81	67	-7	65	-23	58	71	-18	66	-11	46
64	-4	63	-15	60	68	-1	63	-18	64	72	-9	67	-6	56
65	-4	70	-14	61	69	-9	77	-16	63	73	-13	70	-8	59
66	-5	56	-15	47	70	-4	76	-18	73	74	-7	74	-18	74

【００８９】

【表１６】

Table 9-34 – Initialisation parameters for context identifiers *ctx_cbp4, ctx_sig, ctx_last, ctx_abs_level* for context category 0 – 4

Context label	Context category 0		Context label	Context category 1		Context label	Context category 2				Context label	Context category 3		Context label	Context category 4	
							I slices		P, B slices							
	m	n		m	n		m	n	m	n		m	n		m	n
ctx_cbp4																
75	-4	72	79	-4	37	83	-2	52	-4	57	87	0	55	91	-3	41
76	-4	68	80	-3	50	84	-7	68	-9	66	88	-3	70	92	-4	62
77	-6	75	81	-6	49	85	-5	61	-7	64	89	-4	68	93	-6	58
78	-6	75	82	-3	61	86	-8	77	-17	80	90	-4	75	94	-8	73
ctx_sig																
95	-6	68				124	-7	67	4	46	139	-3	71			
96	-11	66	110	0	44	125	-11	64	1	43	140	-9	69	142	-6	60
97	-5	63	111	2	53	126	-8	66	2	45	141	0	70	143	0	63
98	-5	56	112	0	49	127	-11	63	-4	46				144	-3	54
99	2	43	113	1	43	128	-9	63	-1	45				145	-4	54
100	1	47	114	4	45	129	-8	60	3	43				146	4	52
101	-8	58	115	-2	40	130	-11	55	-6	44				147	-5	44
102	-3	46	116	-1	45	131	-10	61	1	45				148	-1	48
103	4	38	117	0	50	132	-7	63	2	46				149	-7	57
104	0	58	118	2	55	133	-7	60	0	46				150	11	51
105	1	51	119	-7	52	134	-16	61	-12	49				151	-13	51
106	-7	57	120	-2	57	135	-2	62	3	50				152	7	55
107	-1	53	121	7	50	136	-1	58	11	46				153	5	57
108	2	47	122	2	52	137	-8	61	4	50				154	2	51
109	-1	59	123	4	66	138	-3	68	9	64				155	4	68
ctx_last																
156	0	5				185	11	25	16	27	200	12	27			
157	1	1	171	4	42	186	9	24	21	19	201	12	28	203	10	28
158	2	2	172	5	46	187	12	24	20	23	202	16	38	204	14	30
159	3	6	173	9	40	188	14	23	21	22				205	17	30
160	4	3	174	7	41	189	13	23	21	23				206	20	30
161	5	4	175	6	46	190	16	22	24	23				207	15	37
162	6	4	176	10	40	191	19	20	25	24				208	21	39
163	7	3	177	14	33	192	18	21	23	27				209	22	33
164	8	4	178	10	43	193	21	21	25	29				210	21	39
165	9	5	179	12	48	194	23	25	23	35				211	15	52
166	10	9	180	13	39	195	20	23	19	36				212	8	49
167	11	2	181	13	41	196	24	25	21	40				213	13	52
168	12	3	182	16	43	197	25	29	23	45				214	8	60
169	13	1	183	21	35	198	24	33	15	53				215	15	56
170	14	6	184	11	55	199	14	53	8	70				216	3	71
ctx_abs_level																
217	-5	55	227	-10	57	237	-10	63	-6	51	247	-9	70	257	-7	58
218	-3	36	228	-1	30	238	-5	37	-5	24	248	-14	55	258	0	33
219	-1	35	229	0	32	239	-7	43	-7	32	249	-10	57	259	-1	40
220	-2	40	230	-1	35	240	-6	46	-4	34	250	-5	56	260	-2	45
221	-6	50	231	0	40	241	-5	49	-5	39	251	-4	57	261	-3	49
222	-3	44	232	-5	39	242	-8	50	-12	43	252	-14	63	262	-7	48
223	-4	51	233	-4	47	243	-7	56	-7	50	253	-11	67	263	-9	58
224	-3	53	234	-9	55	244	-9	62	0	48	254	-5	68	264	-16	66
225	-4	55	235	-6	58	245	-9	64	-3	53	255	-9	71	265	-12	65
226	-11	63	236	-4	56	246	-11	70	-8	60	256	0	50	266	-12	68

【００９０】

9.2.4 Table-based arithmetic coding

注- 算術符号は区間分割を行うことによって符号化を行う。与えられた’０’と’１’の

予測確率p(‘０’)とp(‘１’)=1-p(‘０’)を用い、最初に与えられた区間Rはp(‘０’)´RとR-p(‘０’)´R,にそれぞれ分割される。受け取ったbinary値によって、どちらの区間を次の分割対象区間とするのかが決められる。Binary値は’０’,’１’よりも優勢シンボル(MPS)であるか劣勢シンボル(LPS)であるかが重要であり、それぞれのcontext model CTXはLPSの確率p_{LPS}とMPSの種別(‘０’または’１’)によって決定される。

【００９１】

このRecommendationもしくはInternational Standardにおける算術符号エンジンは以下に示すような3つの特徴を持つ。

【００９２】

確率予測は64の状態を持つstate machineによって行われる。State machineは64の異なるLPSの発生確率(p_{LPS})に対応する状態{P_k | 0≦k＜64}を表を元にして遷移する。

【００９３】

分割区間を表す変数 R は新しい分割区間を計算する前に4つの値{Q_1,…,Q_4}に量子化される。あらかじめQ_i´ P_kに対応する64x4種類の値を計算して保持しておくことでR´ P_kの乗算処理を省くことができる。

‘０’と’１’の発生確率がほぼ等しいとみなせるsyntax elementについては別の復号処理が適用される。

【００９４】

9.2.4.2 Probability estimation

確率予測はLPSの確率{P_k | 0≦k＜64}と遷移法則からなるfinite-state machine (FSM)によって行われる。Table 9-35に与えられたMPSまたはLPSに対する遷移法則を示す。任意のStateからMPSまたはLPSを符号化することによってNext_State_MPS(State)またはNext_State_LPS(State)に遷移する。

【００９５】

状態番号はState=0がLPSの発生確率=0.5に対応し、以下、番号が増えるごとにLPSの発生確率が低くなっているものに対応する。I sliceに対しては状態を最初の24個に制限するため、Table 9-35ではその為に用いられるNext_State_MPS_INTRAがある。尚、Next_State_MPS_INTRAとNext_State_MPSが違うのは一箇所だけである。

【００９６】

I sliceの復号の際23より大きな状態への遷移を避けるため、Next_State_MPS(35)= 23を用いる。詳細はTable 9-35を参照のこと。

1シンボルの符号化/復号を行うたびにStateが変わり、その結果確率が更新される。

```
if(decision = = MPS)
     State←Next_State_MPS_INTRA(State)
else
    State←Next_State_LPS(State)
and all other
slice types
if(decision = = MPS)
    State←Next_State_MPS(State)
else
    State← Next_State_LPS(State).
```

LPSの発生確率が0.5、つまり、State=0の時にさらにLPSが得られた場合、LPSとMPSに対応するシンボル’０’,’１’の交換が行われる。

【００９７】

【表１７】

Table 9-35 – Probability transition

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

【００９８】

【表１８】

Table 9-36 – RTAB[State][Q] table for interval subdivision

State	0	1	2	3	State	0	1	2	3
0	9216	11264	13312	15360	32	896	1152	1344	1536
1	8832	10816	12800	14720	33	896	1088	1280	1472
2	8512	10368	12288	14144	34	832	1024	1216	1408
3	8128	9920	11712	13504	35	832	960	1152	1344
4	7680	9344	11072	12736	36	768	960	1088	1280
5	7168	8768	10368	11968	37	768	896	1088	1216
6	6912	8448	9984	11520	38	704	896	1024	1152
7	6336	7808	9216	10624	39	704	832	960	1152
8	5888	7232	8512	9856	40	640	832	960	1088
9	5440	6656	7872	9088	41	640	768	896	1088
10	5120	6208	7360	8512	42	640	768	896	1024
11	4608	5632	6656	7680	43	576	704	832	960
12	4224	5184	6144	7104	44	576	704	832	960
13	3968	4800	5696	6592	45	576	704	832	960
14	3712	4480	5312	6144	46	512	640	768	896
15	3456	4224	4992	5760	47	512	640	768	896
16	3072	3776	4416	5120	48	512	640	768	832
17	2816	3456	4096	4736	49	512	640	704	832
18	2624	3200	3776	4416	50	512	576	704	832
19	2432	3008	3520	4096	51	448	576	704	768
20	2304	2816	3328	3840	52	448	576	640	768
21	2048	2496	2944	3392	53	448	512	640	704
22	1856	2240	2688	3072	54	448	512	640	704
23	1664	2048	2432	2816	55	448	512	576	704
24	1536	1856	2240	2560	56	384	512	576	704
25	1408	1728	2048	2368	57	384	512	576	640
26	1344	1600	1920	2176	58	384	448	576	640
27	1216	1472	1792	2048	59	384	448	576	640
28	1152	1408	1664	1920	60	384	448	512	640
29	1088	1344	1536	1792	61	384	448	512	640
30	1024	1280	1472	1728	62	384	448	512	576
31	960	1216	1408	1600	63	384	448	512	576

【００９９】

9.2.4.3 Description of the arithmetic decoding engine

算術符号の復号器の状態は、範囲Rの分割区間中を指す値 V によって表される。Figure 9-3に復号処理全体を示す。最初に9.2.4.3.1で規定されるInitDecoder処理を行うことによりVとRが初期化される。1回のdecisionは以下のような2ステップの処理で行われる。最初にcontext model CTXが9.2.2節で示される規則によって生成され、次に与えられたCTXに従って9.2.4.3.2節で規定されるDecode(CTX)が適用されてsymbol S が得られる。

【００１００】

9.2.4.3.1 Initialisation of the decoding engine

Figure 9-4に示される初期化処理において、Vには9.2.4.3.4節で示されるGetByte処理を用いて得られる2 byteの値が設定され、Rには0x8000が設定される。

【００１０１】

9.2.4.3.2 Decoding a decision

Figure 9-5は1つのdecisionを行う処理flowchartを表す。最初のステップではLPSとMPSに対応する区間R_{LPS} とR_{MPS}が以下のように予測される。

与えられた区間の幅Rは、以下に示すように最初にQという値に量子化される。

Q=(R-0x4001)＞＞12, (9-11)

このQとStateを用いてRTABをindexingすることで以下のようにR_{LPS}が得られる。

R_{LPS}=RTAB[State][Q]. (9-12)

Table 9-36 にRTABを16bitで表現したものを示す。RTABは実際には8bitの精度で与えられているが、6bitの左シフトが施され形で与えられている。これは16bitアーキテクチャ

での実装を容易にするためである。
【０１０２】
次に、VとMPS区間R_{MPS}との比較が行われ、V がR_{MPS} 以上であった場合LPSが復号される。同時にVからはR_{MPS}が減ぜられ、RにはR_{LPS}が入る。V がR_{MPS} 未満であった場合、MPSが復号され、RにはR_{MPS}が入る。それぞれのdecoding decisionによって、9.2.4.2節で規定される確率の更新が行われる。新しい区間値Rによっては、9.2.4.3.3節で規定されるrenormalizationが適用される。
【０１０３】
9.2.4.3.3 Renormalization in the decoding engine (RenormD)
R enormalization処理をFigure 9-6に示す。区間値Rと0x4000の比較が最初に行われるが、Rが0x4000よりも大きかった場合renormalizationは行われず、処理は終了する。そうでない場合、renormalization loopに入る。このloop内ではRの値が2倍、つまり、1bit左にshiftされ、bit-count BGは1減らされる。BG＜0となった場合はGetByteによって新たなデータが読み込まれ、Bの最下位ビットがVに設定される。
【０１０４】
9.2.4.3.4 Input of compressed bytes (GetByte)
Figure 9-7に圧縮データの入力処理を示す。初期化時またはrenormalizationでbit-counter BGが負の値になった場合にこの処理が適用される。最初にbitstream Cから新しいbyteが読み込まれる。次に、bitstream中の位置を示すCLの値が1増やされ、bit-counterの値が7に設定される。
【０１０５】
9.2.4.3.5 Decoder bypass for decisions with uniform pdf (Decode_eq_prob)
この処理は動きベクトルの符号、及び、変換係数の符号のように符号化symbolが等しい発生確率を持っているとみなされる場合に適用される特別な処理である。そのような場合、symbol Sを復号し、区間を分割するという通常の処理は一回の比較(V＞=R_{half}?)のみで行える。RenormalizationはFigure 9-8で示されたものと似ているが以下の2点が異なる。第1点目はrescaling処理R←(R＜＜1)が不要なことであり、2点目は、最初の比較(R<=0x4000?)が省けることである。
【先行技術文献】
【特許文献】
【０１０６】
【非特許文献１】Ｆｉｎａｌ Ｃｏｍｍｉｔｔｅｅ Ｄｒａｆｔ ＩＳＯ／ＩＥＣ 14496-10：2002（第９．２節）
【発明の概要】
【発明が解決しようとする課題】
【０１０７】
ところで、画像情報符号化装置１００において、１ピクチャを符号化する際に、その符号化単位を、１ピクチャ、又は、スライス、又は、マクロブロック、又はブロックのいずれとして考えた場合でも、その符号化単位に含まれ、図１１のbinarization器１３１に入力されるシンボルの数は固定ではなく、入力される画像信号や符号化条件によって異なるため、不定である。
【０１０８】
また、binarization器１３１に入力された１シンボルに対して出力される２値データ列の長さは、ＪＶＴ ＦＣＤ第９．２．１節の引用でも示した様に、一定の長さにはならない。例えば、ＪＶＴ ＦＣＤ第９．２．１．５節 Ｔａｂｌｅ９－２０にあるように、Ｉsliceにおけるmb_type１Symbolに対する２値データ列の長さは、最小で１（Intra_4x4時）、最大で６となる。このように、１つのSymbolに対するbinazization器１３１の出力２値データ長も不定である。
【０１０９】
このことから、入力画像信号のある符号化単位に含まれるシンボルに対して、binariza

tion器１３１で出力される２値データの数は固定ではなく不定となり、入力データと符号化条件によっては、非常に大量の２値データがbinarization器１３１から出力される可能性がある。

【０１１０】

ここで、binarization器１３１から出力された２値データは図１１のＣＡＢＡＣ符号化器１３３に入力されることになるが、ＣＡＢＡＣ符号化器１３３は、入力された１個の２値データを処理するのに実装上１クロック以上の処理時間が必要であるため、ＣＡＢＡＣ符号化器１３３に入力される２値データの個数が膨大であると、それだけ処理時間が必要になり、実装した際に、膨大な処理時間がかかることになる。また、上記で述べたように、ＣＡＢＡＣ符号化器１３３に入力される２値データの数は不定であるため、処理時間の最悪値を見積もることが困難になってくる。

【０１１１】

このため、画像情報符号化装置１００において、実時間処理やある一定の処理速度を保証する必要がある場合には、ＣＡＢＡＣ符号化器１３３に入力される２値データの数が膨大であるか、または、不定であると、その保証が不可能となる。

【０１１２】

また、binarization器１３１に入力された１シンボルに対して出力された２値データ列に対するＣＡＢＡＣ符号化器１３３の出力ビット長は不定である。これはＣＡＢＡＣが入力２値データの発生確率に応じて、出力ビット長を可変にコントロールするからである。このため、ＣＡＢＡＣ符号化器１３３に入力された１つの２値データは、その発生確率によって、１ビット以下のビットストリームデータともなり得るし、数ビット以上のビットストリームデータともなり得る。

【０１１３】

ここで、ＣＡＢＡＣ符号化器１３３は、出力される１個のビットデータを処理するのに実装上１クロック以上の処理時間が必要であるため、ＣＡＢＡＣ符号化器１３３から出力されるビットデータが膨大であると、それだけ処理時間が必要になり、実装した際に、膨大な処理時間がかかることになる。また、上記で述べた様に、ＣＡＢＡＣ符号化器１３３から出力されるビットデータの数は不定であるため、処理時間の最悪値を見積もることが困難になってくる。

【０１１４】

このため、画像情報符号化装置１００において、実時間処理やある一定の処理時間を保証する必要がある場合には、ＣＡＢＡＣ符号化器１３３から出力されるビットデータの数が膨大であるか、または、不定であると、その保証が不可能となる。

【０１１５】

以上のように、ＣＡＢＡＣ符号化器１３３へ入出力される２値データやビットデータの個数が、１ピクチャや、ピクチャ内のスライスや、マクロブロックやブロックといった符号化単位内において、不定であり、その数が膨大になり得るというのは、実装上、その符号化単位で、ある一定の処理時間を保証するのを妨げることになる。

【０１１６】

続いて、画像情報復号化装置１２０において、１ピクチャを符号化する際に、その符号化単位を、１ピクチャ、又は、スライス、又は、マクロブロック、又はブロックのいずれとして考えた場合でも、その符号化単位に含まれ、図１４のＣＡＢＡＣ復号化器１６１に入力されるビットストリームのビット数は固定ではなく、入力されるビットストリームによって異なるため、不定である。

【０１１７】

ここで、ＣＡＢＡＣ復号化器１６１は、入力される１個のビットデータを処理するのに実装上１クロック以上の処理時間が必要であるため、ＣＡＢＡＣ復号化器１６１へ入力されるビットデータが膨大であると、それだけ処理時間が必要になり、実装した際に、膨大な処理時間がかかることになる。また、上記で述べた様に、ＣＡＢＡＣ復号化器１６１へ入力されるビットデータの数は不定であるため、処理速度の最悪値を見積もることが困難

になってくる。

【０１１８】

このため、画像情報復号化装置１２０において、実時間処理やある一定の処理時間を保証する必要がある場合には、ＣＡＢＡＣ復号化器１６１へ入力されるビットデータの数が膨大であるか、または、不定であると、その保証が不可能となる。特に、画像情報復号化装置１２０は、画像情報符号化装置１００よりも画像情報をリアルタイムで復号化、表示を行わなければならないという要求が高いため、実時間処理を保証できないことは問題となる。

【課題を解決するための手段】

【０１１９】

かかる課題を解決するため本発明は、画像データを符号化処理する符号化装置において、画像データに対してコンテキストを用いた算術符号化処理を行う符号化手段と、画像データを符号化処理する際の単位であるブロックを非圧縮データのブロックとする場合に、コンテキストを用いた算術符号化処理の終端処理を行うように、符号化手段を制御する制御手段とを備える。

また本発明は、画像データを符号化処理する符号化方法において、画像データに対してコンテキストを用いた算術符号化処理を行う符号化ステップと、画像データを符号化処理する際の単位であるブロックを非圧縮データのブロックとする場合に、コンテキストを用いた算術符号化処理の終端処理を行うように、符号化ステップの符号化処理を制御する制御ステップとを含む。

【発明の効果】

【０１２０】

非圧縮データの符号化処理する際の単位であるブロックに対してコンテキストを用いた算術符号化処理の終端処理を行い、無駄な符号化処理を実行せずに実現することが可能となる。

【図面の簡単な説明】

【０１２１】

【図１】本発明のおける画像情報符号化装置の構成例（装置１０）
【図２】本発明のおける画像情報符号化装置の構成例（装置３０）
【図３】本発明のおける画像情報符号化装置の構成例（装置４０）
【図４】本発明のおける画像情報符号化装置の構成例（装置５０）
【図５】本発明のおける画像情報符号化装置の構成例（装置６０）
【図６】本発明のおける画像情報復号化装置の構成例（装置８０）
【図７】本発明のおけるマクロブロック処理部の構成例
【図８】従来の画像情報符号化装置の構成例
【図９】従来の画像情報復号化装置の構成例
【図１０】ＪＶＴ（従来）における可変長符号化器の構成例
【図１１】ＪＶＴ（従来）におけるＣＡＢＡＣ符号化器の構成例
【図１２】ＪＶＴ（従来）におけるＣＡＶＬＣ符号化器の構成例
【図１３】ＪＶＴ（従来）における可変長復号化器の構成例
【図１４】ＪＶＴ（従来）におけるＣＡＢＡＣ復号化器の構成例
【図１５】ＪＶＴ（従来）におけるＣＡＶＬＣ復号化器の構成例
【図１６】Overview of the Decoding Process
【図１７】Flowchart of initialisation of the decoding engine
【図１８】Flowchart for decoding a decision
【図１９】Flowchart of renormalization
【図２０】Flowchart for Input of Compressed Bytes
【図２１】Flowchart of decoding bypass
【図２２】Illustration of the generic context template using two neighbouring sy

mbols A and B for conditional coding of a current symbol C

【発明を実施するための形態】

【０１２２】

以下、本発明の実施例を図面を参照しながら説明する。

【０１２３】

図１における装置１０による実施例の説明

本発明における画像符号化装置の実施例を図１に示す。図１の装置１０では符号化されるべき画像信号が入力され、符号化されたビットストリームが出力される。装置１０は入力バッファ１１と変換処理部１２とＣＡＢＡＣ処理部１３と制限監視器１４と出力バッファ１５により構成される。入力バッファ１１では、入力画像をマクロブロック単位に分割し出力をし、後段においてマクロブロックの処理が終わる毎に、次のマクロブロックを出力する。変換処理部１２では、入力されたマクロブロック画像に対して処理を行い、ヘッダ情報や量子化された係数情報をＣＡＢＡＣ処理部１３に出力する。具体的には、パラメータ設定器１６により、マクロブロックのモード情報や動きベクトル情報、量子化パラメータ等のヘッダ情報を設定し、その値（シンボル）を予測器１７、ＤＣＴ器１８、量子化器１９、及び、ＣＡＢＡＣ処理部１３に出力する。ここで、パラメータ設定器１６は、マクロブロックのヘッダ情報のみならず、スライスやピクチャのヘッダ情報も設定し、出力することを可能とするので、ここでは、全てを総称してヘッダ情報と記載する。なお、予測器１７では動き補償が、ＤＣＴ器１８ではＤＣＴ変換が、量子化器１９では量子化処理が、それぞれ、前段からの入力信号に対して、パラメータ設定器１６からの入力信号を参照して適用される。

【０１２４】

ＣＡＢＡＣ処理部１３には、ヘッダ情報と量子化された係数情報がシンボルデータとして入力され、算術符号化が適用され、ビットデータとして出力される。具体的には、入力されたシンボルデータをbinarization器２０によって２値データ列に変換し、その２値データをContext演算器２１からのContext情報をもとに、ＣＡＢＡＣ符号化器２２でエントロピー符号化する。Context演算器２１では、binarization器２０に入力されるシンボルデータと、binarization器２０から出力される２値データをもとにContextを更新し、また、そのContext情報をＣＡＢＡＣ符号化部２２に出力する。

【０１２５】

制限監視器１４は、ＣＡＢＡＣ符号化器２２に入力される２値データの個数のカウンタと出力されるビットデータの個数のカウンタ（ビットカウンタ２５）をそれぞれ独立に持ち、ＣＡＢＡＣ符号化器２２へ２値データが入力されるたびに前者のカウンタを１つずつ増加させ、ＣＡＢＡＣ符号化器２２からビットデータが出力されるたびに後者のカウンタを１つずつ増加させる。このカウンタはそれぞれ、マクロブロックの先頭の処理を開始するたびに０にリセットされる。これにより、各マクロブロックにおける、ＣＡＢＡＣ符号化器２２の入力データと出力データのそれぞれの個数をカウントすることが可能となる。

【０１２６】

この制限監視器１４では、これらカウンタのうちのどちらか一方でも、あらかじめ設定された閾値を超えてしまった場合、その符号化データは無効であることを示す信号（以下、再符号化信号）を、出力バッファ１５とContext演算器２１とパラメータ設定器１６に対して出力する。この再符号化信号を受け取ったパラメータ設定器１６は、再度この閾値を超えない様に注意して符号化パラメータを設定し直し、符号化対象のマクロブロックデータを再符号化処理する。また、Context演算器２１は、Contextメモリ群２３を有しており、このContextメモリ群２３は従来の技術の図１０３にある従来のContextメモリ群１３５と同様に、符号化処理中に随時更新されるContextとリセット時などに用いられるContextの初期状態が保存されるが、それと共に、マクロブロックをデータ処理する直前のContextの状態も保存しておくことが可能である。これにより、再符号化信号を受け取ったContext演算器１６は、内部のContextの状態を、この新たに付け加えられたメモリに保存されているContextの値に書き換えることにより、符号化対象のマクロブロックのデータに

よって更新される直前のContextの状態に復元することが可能である。また、再符号化信号を受け取った出力バッファ１５は、内部に蓄積された符号化対象マクロブロックのビットデータを全て削除し、新たな符号化パラメータで符号化されたマクロブロックデータの入力を待つ。逆に、対象のマクロブロックの符号化処理を終えた際に、制限監視器１４のいずれのカウンタも、あらかじめ設定された閾値を超えていなければ、出力バッファ１５内の対象マクロブロックのビットデータをビットストリームとして出力することが可能である。

【０１２７】

ここまで説明した図１の装置１０では、制限監視器１４にあるカウンタはマクロブロックの先頭でリセットされることから、これは、マクロブロック単位で、ＣＡＢＡＣ符号化器２２に入力される２値データと、出力されるビットデータの個数を監視し制限することを意味するが、このリセットのタイミングを、マクロブロック内のブロック単位に設定すれば、各ブロック単位での上記データ個数を監視し制限することが可能になる。同様にスライス単位でリセットをすれば、スライス単位で上記データ個数を監視し制限することが可能になるし、ピクチャ単位でリセットすれば、ピクチャ単位で上記データ個数を監視し制限することが可能となる。また、この様に上記データ個数を監視、制限する符号化単位を変える場合には、同時に、Context演算器２１におけるContextメモリ群２３にはその符号化単位の直前のContext値が保存されるため、Contextの状態復元も、その符号化単位の直前の状態に復元されることになる。また、出力バッファ１５のビットデータの削除も同様の符号化単位で行われることになる。

【０１２８】

なお、この復元の際には、Contextメモリ２３群に保存されている符号化単位の直前のContext値ではなく、同様にContextメモリ群２３に保存されているあらかじめ定められた初期値に復元することも可能である。

【０１２９】

ここまで説明した図１の装置１０では、制限監視器１４には２つのカウンタが設定されているが、これらカウンタに設定する閾値は、それぞれ独立に自由な値を設定することが可能であり、また、２つのうちどちらか一方のみに対応するデータカウントのみを監視し、もう片方は無視する、もしくはカウンタ自体を持たない様な構成にすることも可能である。

【０１３０】

この装置１０により、１回のマクロブロック処理において、ＣＡＢＡＣ符号化器に入力、及び出力されるデータ量の上限を制限することができるため、要求されたマクロブロック１回の処理時間を満たすことが可能になる。また、要求された処理時間で復号化処理が可能なビットストリームを出力することが可能になる。

【０１３１】

ここで、今後の説明のために、図１における変換処理部１２とＣＡＢＡＣ処理部１３をまとめて表現した装置、マクロブロック処理部２９を図７に示す。今後の記載におけるマクロブロック処理部２９は、図１における変換処理部１２とＣＡＢＡＣ処理部１３を直列でつないだ装置として、同様の振る舞いをするものとする。

【０１３２】

図２における装置３０による実施例の説明

図１で示した装置１０では、制限監視器１４が再符号化信号を出力するたびに、変換処理部１２は再度、新たな符号化パラメータを設定して対象マクロブロックの符号化を行わなければならないし、再度設定されたパラメータによって得られたデータにより、再度、制限監視器１４のカウンタが閾値を超えてしまうということが繰り返される可能性もある。そのため、１つのマクロブロックに対して複数回、符号化処理を連続的に適用しなければならず、その分、１ピクチャの符号化時間が多くかかってしまう。

【０１３３】

ゆえに、本発明における画像符号化装置の別の実施例として、対象マクロブロックに対

して異なる符号化パラメータを適用する符号化を並列に実施する例を２番目の実施例として図２に示す。

【０１３４】

図２の装置３０では、図１の装置１０と同様に、符号化されるべき画像信号が入力され、符号化されたビットストリームが出力される。図２の装置３０は入力バッファ３１と、Ｎ個の異なる符号化パラメータによるＮ段の並列符号化処理を可能とするマクロブロック処理部３２－１～３２－Ｎと、それに対応した出力バッファ３３－１～３３－Ｎと、制限監視・経路選択器３４と切替器３５により構成される。

【０１３５】

図２の装置３０では、符号化対象のマクロブロックに対して、Ｎ個の異なる符号化パラメータを設定し、それぞれの符号化パラメータによる符号化処理をマクロブロック処理部３２－１～３２－Ｎで並列に行い、その出力を出力バッファ３３－１～３３－Ｎに蓄積する。

【０１３６】

制限監視・経路選択器３４は、マクロブロック処理部３２－１～３２－Ｎそれぞれに対応するＣＡＢＡＣ符号化器に対する２つの入出力データカウンタ（ビットカウンタ３６）を持ち、Ｎ段の並列経路のうちから、このカウンタが閾値を超えず、かつ、符号化効率の最も優れた符号化経路を選択し、切替器３５により出力する系統を選択する。

【０１３７】

図２の装置３０におけるその他の詳細な動作、及び符号化単位等のバリエーションは図１の装置１０と同様のものとする。

【０１３８】

この装置３０により、符号化時に置いてＣＡＢＡＣ符号化器に入力、及び出力されるデータ量の上限を制限することができるため、要求されたの符号化処理時間を満たすことが可能になる。また、要求された処理時間で復号化処理が可能なビットストリームを出力することが可能になる。

【０１３９】

図３における装置４０による実施例の説明

次に図３に、また別の本発明における画像符号化装置の実施例を示す。この実施例は、図１の実施例に加えて、非圧縮符号化データ、すなわち、入力されたマクロブロックに対して、圧縮しないＲＡＷデータをそのまま符号化する経路を有している。

【０１４０】

図３の装置４０の図１の装置１０との違いは、マクロブロック画像データはマクロブロック処理部４１のみでなく、非圧縮符号化部４３にも入力される。非圧縮符号化部４３では、入力された画像情報に対して一切の変換処理とエントロピー符号化をしないデータ、すなわち、ＲＡＷデータが出力バッファＢ４４に出力される。制限監視・経路選択器４７は、図１における制限監視器１４の振る舞いと同様に、ビットカウンタ４９によりＣＡＢＡＣ符号化器の入出力のデータ量を監視し、かつ、もし、その監視されているデータがあらかじめ設定されている閾値を超えた場合には、切替器４６が出力バッファＢ４４からの入力を選択し出力するようにする。逆に、閾値を超えなかった場合には、出力バッファＡ４２、出力バッファＢ４４のどちらの出力でも選択することが可能となっている。

【０１４１】

ここで、制限監視・経路選択器４５が出力バッファＢ４４、すなわちＲＡＷデータを選択した場合には、その事がマクロブロック処理部４１にあるContext演算器にも通知され、Context演算器内のContext値は、Contextメモリ群に保存されたマクロブロックを処理する直前のContextの値を用いて、そのＲＡＷデータとして処理されたマクロブロックを処理する直前の状態に復元される

【０１４２】

なお、マクロブロックがＲＡＷデータとして処理された場合のContextの復元方法としては、あらかじめ決められた初期状態に復元するということも可能である。

【０１４３】
ここで、マクロブロックがＲＡＷデータとして符号化されたかどうかを示すために、出力されるビットストリームのヘッダ情報には、そのためのデータが埋め込まれる。
【０１４４】
ＲＡＷデータが符号化された際には、ＣＡＢＡＣ符号化部は、ＲＡＷデータをビットストリームに出力させる前に、ＣＡＢＡＣの終端処理をする。
【０１４５】
また、非圧縮符号化器４５は、ＲＡＷデータを出力する非圧縮処理装置のみでなく、ＤＰＣＭ符号化装置などさまざまな圧縮装置に置き換えることが可能である。
【０１４６】
図３の装置４０におけるその他の詳細な動作、及び符号化単位等のバリエーションは図１の装置１０と同様のものとする。
【０１４７】
この装置４０により、符号化時に置いてＣＡＢＡＣ符号化器に入力、及び出力されるデータ量の上限を制限することができるため、要求されたの符号化処理時間を満たすことが可能になる。また、要求された処理時間で復号化処理が可能なビットストリームを出力することが可能になる。
【０１４８】
図４における装置５０による実施例の説明
次に図４に、また別の本発明における画像符号化装置の実施例である装置５０を示す。この実施例は、図２の装置３０に加えて、非圧縮符号化データ、すなわち、入力されたマクロブロックに対して、圧縮しないＲＡＷデータをそのまま符号化する経路を有している。
【０１４９】
図４の装置の図２の装置との共通の部分の動作は、図１の装置とほとんど同じであるため、違いのみを具体的に示すと、マクロブロック画像データはマクロブロック処理部５１－１～５１－Ｎのみでなく、非圧縮符号化部５８に入力される。非圧縮符号化部５８では、入力された画像情報に対して一切の変換処理とエントロピー符号化をしないデータ、すなわち、ＲＡＷデータとして出力バッファＢ５９に出力される。制限監視・経路選択器５３は、図２における制限監視・経路選択器３４の振る舞いと同様に、ビットカウンタ５５を監視し、もし、全ての経路１～Ｎにおけるビットカウンタ（２個のうちどちらか）があらかじめ設定されている閾値を超えた場合には、信号選択器５４が出力バッファＢ５９からの入力を選択し出力するようにする。逆に、閾値を超えなかった場合には、出力バッファＡ５２－１～５２－Ｎ、出力バッファＢ５９のどちらの出力でも選択することが可能となっている。
【０１５０】
なお、出力バッファＢ５９からのＲＡＷデータが信号選択部５４で選ばれる場合には、マクロブロック処理部５１－１～５１－NにあるContext演算部のContextの状態は、Contextメモリ群で記憶されたマクロブロックを処理する直前のContextの状態に復元される。なお、この復元の際には、図１の装置１０でも説明したとおり、あらかじめ定められた初期値に復元することも可能である。
【０１５１】
逆に、出力バッファからＲＡＷデータでなく、出力バッファＡ５２－１～５２－Ｎのうちの１つである出力バッファＡ５２－ｉが信号選択部５４で選ばれた場合には、マクロブロック処理部５１－ｉのContext演算部のContextの状態が他のマクロブロック処理部５１－１～５１－NにあるContext演算部にコピーされる。これは、その後のマクロブロックの符号化を始めるにあたって、全てのContext演算部のContextの状態が同じでなければならないからである。
ちなみに、非圧縮符号化部５８は、ＲＡＷデータを出力する非圧縮処理装置のみでなく、ＤＰＣＭ符号化装置などさまざまな圧縮装置に置き換えることが可能である。

【０１５２】
図４の装置５０におけるその他の詳細な動作、及び符号化単位等のバリエーションは図１の装置１０と同様のものとする。
【０１５３】
この装置５０により、符号化時に置いてＣＡＢＡＣ符号化器に入力、及び出力されるデータ量の上限を制限することができるため、要求されたの符号化処理時間を満たすことが可能になる。また、要求された処理時間で復号化処理が可能なビットストリームを出力することが可能になる。
【０１５４】
図５における装置６０による実施例の説明
次に図５に、図１００の可逆符号化部１０６としてＣＡＢＡＣではなくＣＡＶＬＣを適用する装置６０を示す。この装置６０は図１の装置１０のＣＡＢＡＣ処理部１３をＣＡＶＬＣ処理器６３で置き換えたものであり、ＣＡＶＬＣ処理器６３と制限監視器６４以外は同じ振る舞いをするため、ここではＣＡＶＬＣ処理器６３と制限監視器６４の動作についてのみ説明をする。
【０１５５】
ＣＡＶＬＣ処理器６３には、ヘッダ情報と量子化された係数情報がシンボルデータとして入力され、従来のＭＰＥＧ２などと類似した、可変長テーブルを用いた可変長符号化が適用され、ビットデータとして出力される。ここでＣＡＶＬＣ処理器６３は、従来の技術の図１０４で説明したＣＡＶＬＣ符号化器とContext保存器からなり、従来の保存器と同様に、既にＣＡＶＬＣ符号化器で符号化された情報、例えば、処理中のブロックだけでなく既に処理されたブロックにおける各ブロック内の非０係数の個数や直前に符号化された係数の値などが保存されるのに加えて、本発明におけるＣＡＶＬＣ処理器６３では、再符号化信号が来たときにマクロブロックを符号化する直前の状態に戻れる様に、マクロブロックを符号化する直前のContextの状態を保存しておくことが可能である。ＣＡＶＬＣ符号化器は、このContext保存器からの情報をもとにシンボルに適用する可変長符号テーブルを切り替えることが可能である。なお、Context保存器にはリセット時などに用いられるContextの初期状態も保存される。
【０１５６】
制限監視器６４は、ＣＡＶＬＣ処理器６３から出力されるビットデータの個数のカウンタ（ビットカウンタ７５）を１つ持ち、ＣＡＶＬＣ処理器６３からビットデータが出力されるたびにこのカウンタを１つずつ増加させる。このカウンタは、マクロブロックの先頭の処理を開始する時に０にリセットされる。これにより、各マクロブロックにおける、ＣＡＶＬＣ処理機６３からの出力データの個数をカウントすることが可能となる。
【０１５７】
この制限監視器６４では、このカウンタ７５があらかじめ設定された閾値を超えてしまった場合、その符号化データは無効であることを示す信号（以下、再符号化信号）を、出力バッファ６５とパラメータ設定器６６に対して出力する。この再符号化信号を受け取ったパラメータ設定器６６は、再度この閾値を超えない様に注意して符号化パラメータを設定し直し、符号化対象のマクロブロックデータを再符号化処理する。また、再符号化信号を受け取った出力バッファ６５は、内部に蓄積された符号化対象マクロブロックのビットデータを全て削除し、新たな符号化パラメータで符号化されたマクロブロックデータの入力を待つ。
【０１５８】
図５の装置６０におけるその他の詳細な動作、及び符号化単位等のバリエーションは図１の装置１０と同様のものとする。
【０１５９】
この装置６０により、１回のマクロブロック処理において、ＣＡＶＬＣ符号化器から出力されるデータ量の上限を制限することができるため、要求されたマクロブロック１回の処理時間を満たすことが可能になる。また、要求された処理時間で復号化処理が可能なビ

ットストリームを出力することが可能になる。

【０１６０】

また、図１の装置のみでなく、図２～図４の装置に対しても、ＣＡＢＡＣ処理部をＣＡＶＬＣ処理部に置き換えることが可能であり、その振る舞いはここで示した実施例と同様なものとする。但し、ＣＡＶＬＣ処理器はマクロブロックがＲＡＷデータとして符号化されると、そのマクロブロックのContextを持てなくなるので、その様なときのためにＲＡＷデータが符号化された際のContextの更新の仕方を定義しておく必要がある。この定義の方法は符号化装置と復号化装置間で同期が取れていればどんなものでも良い。例えば、ＲＡＷデータとして符号化されたマクロブロック内のブロックに存在している非０係数の個数は１５とみなすなどである。その装置により、符号化時においてＣＡＶＬＣ符号化器から出力されるデータ量の上限を制限することができるため、要求されたの符号化処理時間を満たすことが可能になる。また、要求された処理時間で復号化処理が可能なビットストリームを出力することが可能になる。

【０１６１】

図６における装置８０による実施例の説明

次に、図１～４の装置に対応した本発明における画像情報復号化装置である装置８０を図６を用いて示す。なお、図１、２の装置においては、非圧縮符号化部とその経路がないため、図６の装置８０において、非圧縮復号化部８８への経路は選択されることはない。この様なことが明らかな場合には、この非圧縮復号化部８８とその経路を実装しないことも可能である。

【０１６２】

図６の装置８０では復号化されるべきビットストリームが入力され、復号化された画像信号が出力される。図６の装置８０は経路選択器Ａ８１、Ｂ８５と符号化方式判定器８４と逆変換処理部８３とＣＡＢＡＣ処理部８２と制限監視器８６と非圧縮復号化器８８により構成される。

【０１６３】

まず、各マクロブロックを処理する始めは、経路選択器Ａ８１、Ｂ８５はＣＡＢＡＣ処理部８２の経路を選択している。ＣＡＢＡＣ処理部８２では、入力されたビットストリームからマクロブロックを復号化する際に、まず、ビットストリームに埋め込まれた、そのマクロブロックがＲＡＷデータかどうかを示すシンボルを復号化し、それが符号化方式判定器８４でＲＡＷデータであると判定された場合は、経路選択器Ａ８１、Ｂ８５は非圧縮復号化部８８の経路を選択し、非圧縮復号化部８８からの出力を、画像信号として出力する様にする。ここで、非圧縮復号化部８８では固定長復号化を行い画像データを取得する。この非圧縮符号化部８８が選択された場合には、ＣＡＢＡＣ処理部８２のContext演算部９２内のContextの状態は変更しなくても構わないし、あるあらかじめ決められた値で初期化しても良いし、その他の法則を用いて変更しても構わず、符号化装置側のＣＡＢＡＣ処理部の振る舞いと同期が取れていれば良い。また、この際に、同一ピクチャ内の後で復号化されるマクロブロックの復号化に用いられるプレディクタはあらかじめ決められた値に設定される。例えば、非圧縮復号化されたマクロブロックの動きベクトルは０に設定され、マクロブロックタイプはイントラ符号化と言った様に設定される。このプレディクタの値も符号化器側と同期が取れていればどんな値を設定しても良い。

【０１６４】

逆に、符号化方式判定器８４によって、マクロブロックデータがＣＡＢＡＣ処理部８２で処理されることが選択された場合は、引き続き入力ビットストリームはＣＡＢＡＣ処理部８２に入力される。

【０１６５】

ＣＡＢＡＣ処理部８２では、入力ビットストリームから、ヘッダ情報と量子化された係数情報がシンボルデータとして復号化され出力される。具体的には、入力されたビットストリームを、Context演算器９２からのContext情報をもとに、ＣＡＢＡＣ復号化器９０でエントロピー復号化し、そこで出力された２値シンボル列を逆binarization器９１により

、シンボルデータに変換する。Context演算器９２では、逆binarization器９１に入力される２値データと逆binarization器９１から出力されるシンボルデータをもとにContextを更新し、また、そのContext情報をＣＡＢＡＣ復号化部９０に出力する。このＣＡＢＡＣ処理部８８の動作は、「従来の技術」で説明したＪＶＴ
ＦＣＤ第９．２節の記述に準ずるものとする。

【０１６６】

逆変換処理部８３では、入力されたヘッダ情報や量子化された係数情報を、逆量子化、逆ＤＣＴ、動き補償することにより画像信号を復号化し出力する。

【０１６７】

制限監視器８６は、ＣＡＢＡＣ復号化器９０へ入力されるビットデータの個数のカウンタと出力される２値データの個数のカウンタ（ビットカウンタ９３）をそれぞれ独立に持ち、ＣＡＢＡＣ復号化器９０へビットデータが入力されるたびに前者のカウンタを１つずつ増加させ、ＣＡＢＡＣ復号化器９０から２値データが出力されるたびに後者のカウンタを１つずつ増加させる。このカウンタはそれぞれ、マクロブロックの先頭の処理を開始する時に０にリセットされる。これにより、各マクロブロックにおける、ＣＡＢＡＣ復号化器９０における入力データと出力データのそれぞれの個数をカウントすることが可能となる。

【０１６８】

この制限監視部８６では、これらカウンタのうちのどちらか一方が、あらかじめ設定された閾値を超えてしまった場合、エラー処理を実行する。エラー処理としては、復号化処理をいったん中止し、次のスライスヘッダやピクチャヘッダを待って再度、復号化処理を開始したり、単に警告のみを発するだけで復号化処理は引き続き続けるということが可能である。また、エラー処理をせず、復号化処理を引き続き続けるということも可能である。

【０１６９】

この装置８０により、復号化時においてＣＡＢＡＣ復号化器９０に入力、及び出力されるデータ量を監視することができるため、仮にこの上限を超えるデータ量が入出力されたとしても、要求された復号化処理時間を満たすようにエラー処理等を施すことが可能となる。

【０１７０】

また、実装の方法としては、装置８０においては、必ずしも制限監視部８６は実装されるとは限らない。その場合には、ＣＡＢＡＣ符号化器９０において入出力されるデータ量は監視されない。

【０１７１】

なお、装置８０ではエントロピー復号化としてＣＡＢＡＣを適用した際の、本発明における画像情報復号化装置の実施例を示したが、既に、画像符号化装置の実施例でも示したとおり、このＣＡＢＡＣ処理部はＣＡＶＬＣ処理部で置き換えることが可能であり、その実装方法は符号化装置の実施例でも説明した通り、ほとんど１対１で類似しているため、ここでは説明を割愛する。なお、その符号化装置と同様に、マクロブロックがＲＡＷデータで符号化された際のＣＡＶＬＣのContextの更新の仕方をあらかじめ定義しておく。

【０１７２】

本発明におけるビットストリームの実施例の説明

次に、本発明における符号化されたビットストリームの実施例について示す。ここまでの説明でも述べた様に、本発明ではビットストリーム内に圧縮したデータでも、ＲＡＷデータでも符号化することが可能である。このために、本発明のビットストリームでは、マクロブロックヘッダにおいて、そのマクロブロックがＲＡＷデータとして符号化されているかそうでないかを明示的に示すヘッダ情報を付加し、そのヘッダ情報の後にＲＡＷデータ、もしくは、圧縮されたビットデータのどちらかを続けている。ここで、ＲＡＷデータとして符号化されているかそうでないかを明示的に示すのに、マクロブロックヘッダ情報の１つであるmacroblock typeによって明示する。逆に言うと、本発明におけるビットス

トリームは、異なる符号化方式をマクロブロック単位で混在させることが可能であるということである。

【０１７３】

またここでは、マクロブロックのヘッダ情報としてそのマクロブロックの符号化方式を指定する情報が組み込まれている場合を示したが、この指定情報を、スライスヘッダやピクチャヘッダに組み込めば、それらの単位で符号化方式の混在と、その指定を行うことが可能となる。

【０１７４】

ここで、本発明におけるビットストリームには、このヘッダ情報（例えば、macroblock type）がＣＡＢＡＣで符号化され、続いてＲＡＷデータ（すなわち固定長ビット列）が符号化される場合には、ＲＡＷデータを符号化する前に、ＣＡＢＡＣのの終端処理をされたビットが挿入される。

【０１７５】

また、本発明におけるビットストリームはＣＡＢＡＣで符号化された場合には、これまでの実施例でも述べた様に、ＣＡＢＡＣ符号化器、及び復号化器の入力と出力のビットカウンタのどちらか一方でもあらかじめ設定された閾値を超えることのないデータにより構成される。また、ＣＡＶＬＣで符号化された場合には、ＣＡＶＬＣ符号化器の出力、及び復号化器の入力のビットカウンタが、あらかじめ設定された閾値を超えることのないデータにより構成される。

これらのことから、本発明におけるビットストリームにより、画像情報符号化器、及び画像情報復号化器に対して、ある一定の復号化処理時間を保証することを可能としている。

【符号の説明】

【０１７６】

１０、３０、４０、５０、６０、８０……装置、１１、６１……入力バッファ、１２、６２……変換処理部、１３……ＣＡＢＡＣ処理部、２９、３２、４１、５１……マクロブロック処理部、６３……ＣＡＶＬＣ処理器。

【図１】

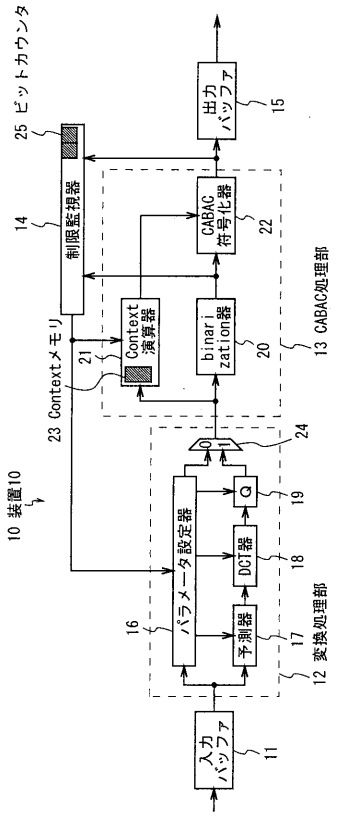

図１ 本発明による画像情報符号化装置の構成

【図２】

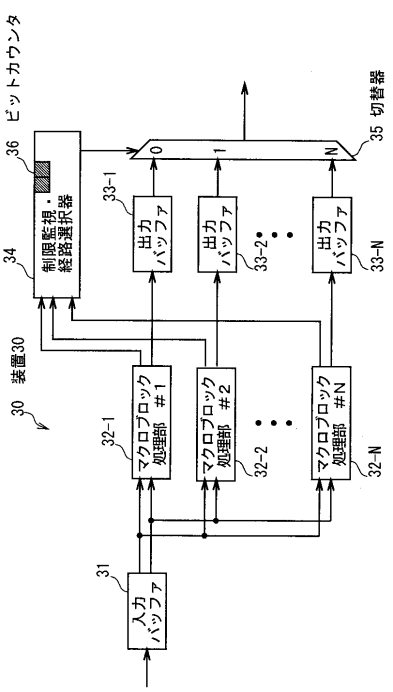

図２ 本発明による画像情報符号化装置の構成

【図３】

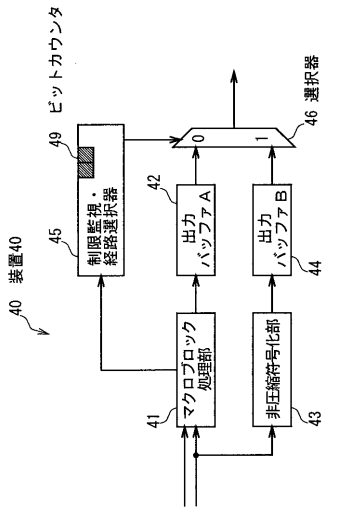

図３ 本発明による画像情報符号化装置の構成

【図４】

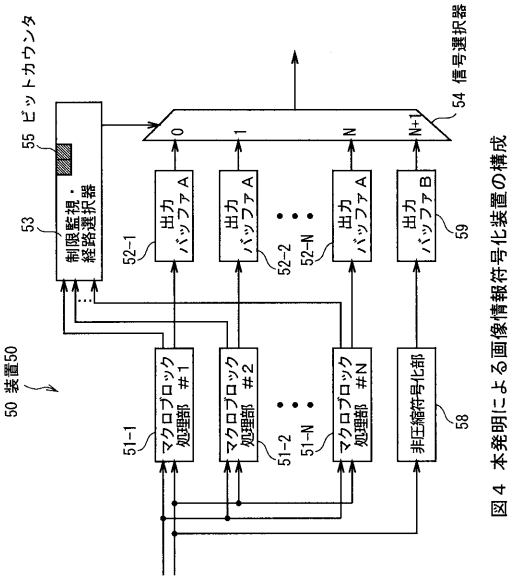

図４ 本発明による画像情報符号化装置の構成

【図５】

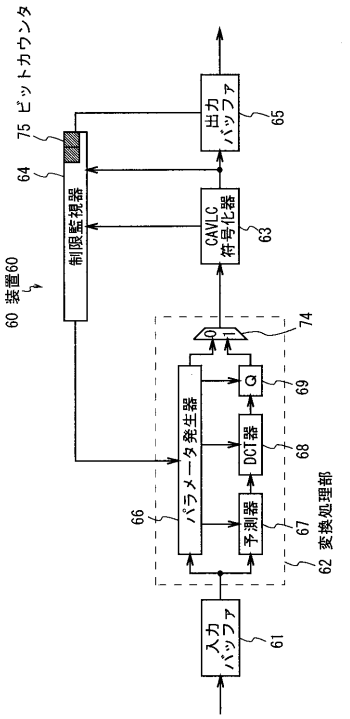

図５ 本発明による画像情報符号化装置の構成

【図６】

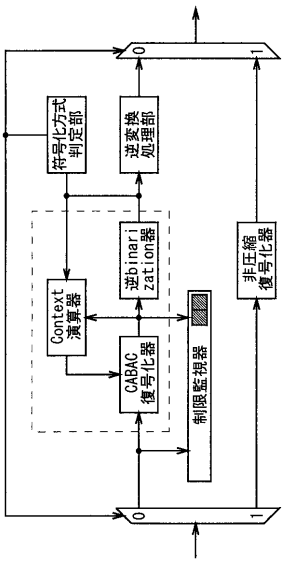

図６ 本発明による画像復号化装置の構成

【図７】

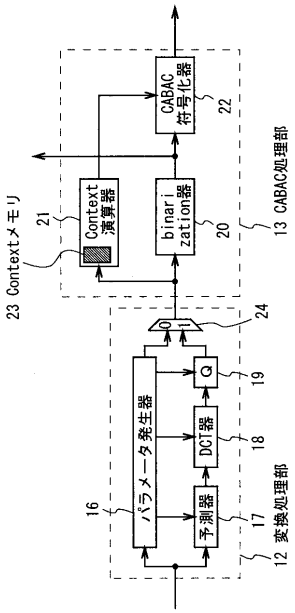

図７ マクロブロック処理部の構成

【図８】

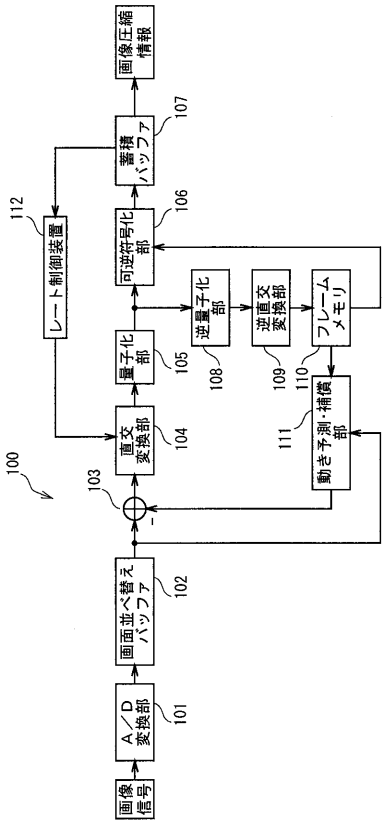

図８ 従来の画像情報符号化装置の構成

【図９】

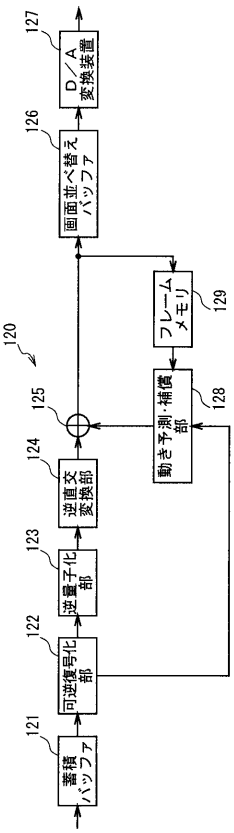

図９ 従来の画像情報復号化装置の構成

【図１０】

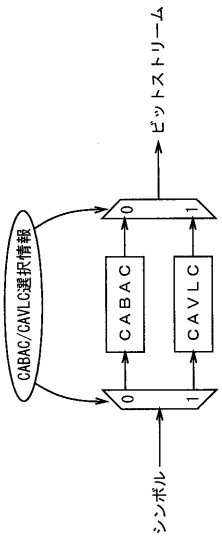

図１０ 従来の可変長符号化器の構成

【図１１】

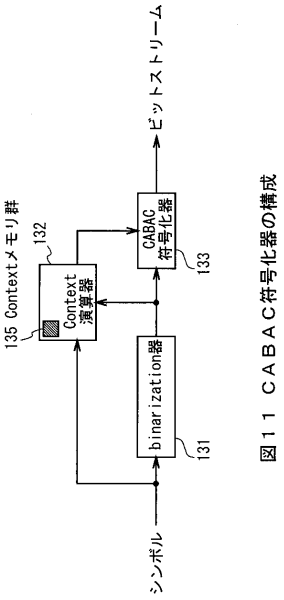

図１１ ＣＡＢＡＣ符号化器の構成

【図１２】

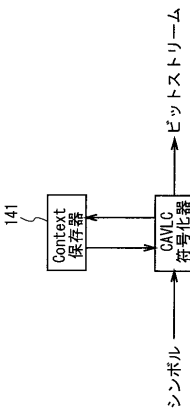

図１２ ＣＡＶＬＣ符号化器の構成

【図１３】

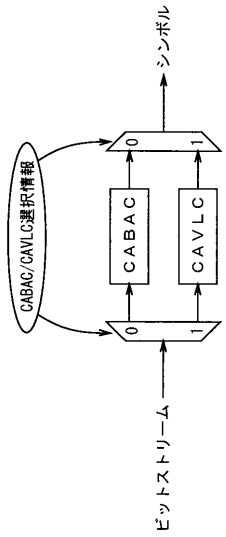

図１３ 従来の可変長復号化器の構成

【図１４】

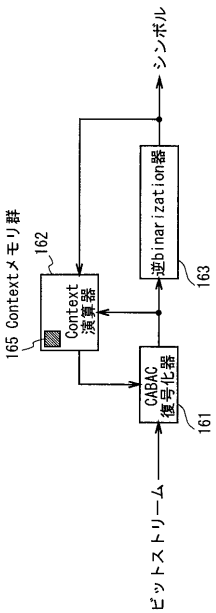

図１４ ＣＡＢＡＣ復号化器の構成

【図１５】

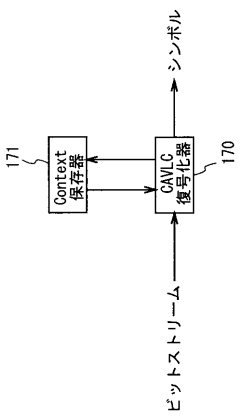

図１５ ＣＡＶＬＣ復号化器の構成

【図１６】

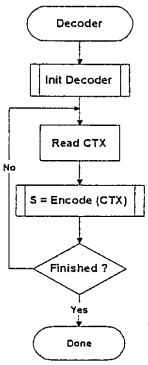

Figure 9-3 - Overview of the Decoding Process

図１６

【図１７】

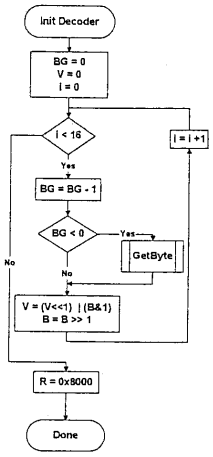

Figure 9-4 – Flowchart of initialisation of the decoding engine

図１７

【図１８】

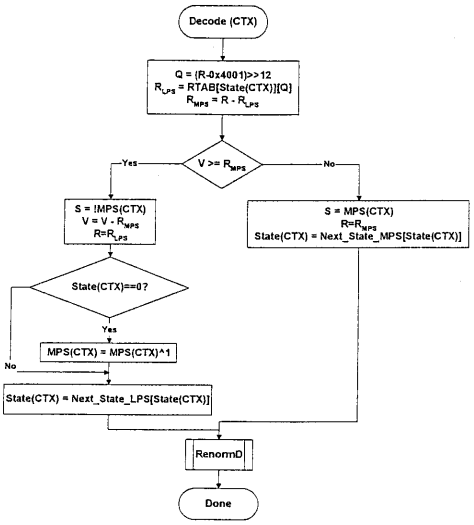

Figure 9-5 – Flowchart for decoding a decision

図１８

【図１９】

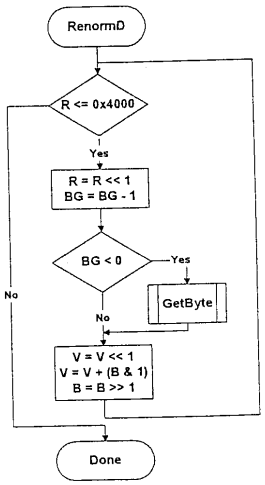

Figure 9-6 – Flowchart of renormalization

図１９

【図２０】

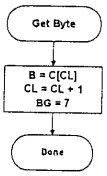

Figure 9-7 – Flowchart for Input of Compressed Bytes

図２０

【図２１】

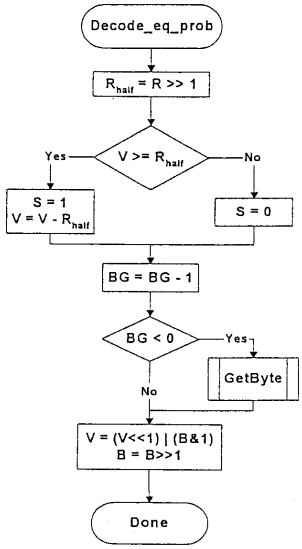

Figure 9-8 – Flowchart of decoding bypass

図２１

【図２２】

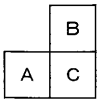

Figure 9-2—Illustration of the generic context template using two neighbouring symbols A and B for conditional coding of a current symbol C

図２２

フロントページの続き

審査官　矢野　光治

(56)参考文献　特開平０９－００９２６１（ＪＰ，Ａ）
特開平１０－１０８１８４（ＪＰ，Ａ）
特開平１０－１９１３４３（ＪＰ，Ａ）
特開平１０－２５６９１７（ＪＰ，Ａ）
特開２００１－２３０９３５（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｈ０４Ｎ　　７／２４－７／６８