

(12) 特許協力条約に基づいて公開された国際出願

(19) 世界知的所有権機関
国際事務局

(43) 国際公開日
2011年10月6日(06.10.2011)



PCT



(10) 国際公開番号
WO 2011/121843 A1

(51) 国際特許分類:
H04N 7/32 (2006.01)

(21) 国際出願番号: PCT/JP2010/071155

(22) 国際出願日: 2010年11月26日(26.11.2010)

(25) 国際出願の言語: 日本語

(26) 国際公開の言語: 日本語

(30) 優先権データ:
PCT/JP2010/055640 2010年3月30日(30.03.2010) JP

(71) 出願人 (米国を除く全ての指定国について): 株式会社 東芝 (KABUSHIKI KAISHA TOSHIBA) [JP/JP]; 〒1058001 東京都港区芝浦一丁目1番1号 Tokyo (JP).

(72) 発明者: および

(75) 発明者/出願人 (米国についてのみ): 竹島 秀則 (TAKESHIMA, Hidenori) [JP/JP]. 浅野 渉 (ASANO, Wataru) [JP/JP]. 谷沢 昭行 (TANIZAWA, Akiyuki) [JP/JP].

(74) 代理人: 蔵田 昌俊, 外 (KURATA, Masatoshi et al.); 〒1050001 東京都港区虎ノ門1丁目12番9号 鈴榮特許総合事務所内 Tokyo (JP).

(81) 指定国 (表示のない限り、全ての種類の国内保護が可能): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

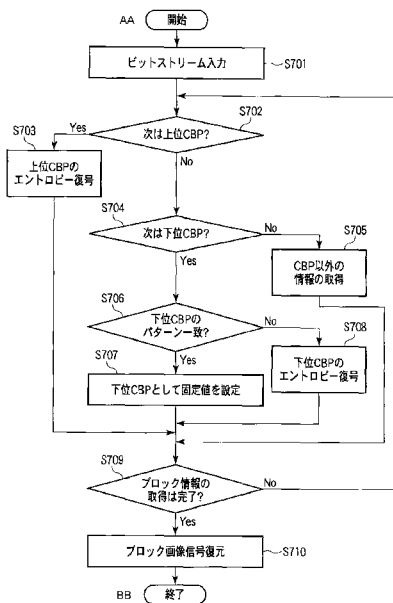
(84) 指定国 (表示のない限り、全ての種類の広域保護が可能): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), ユーラシア (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), ヨーロッパ (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT,

[続葉有]

(54) Title: IMAGE CODING AND IMAGE DECODING METHOD

(54) 発明の名称: 画像符号化及び画像復号化方法

[図7]



AA START
BB END
S701 INPUT BIT STREAM
S702 IS TOP CBP NEXT?
S703 ENTROPY DECODE TOP CBP
S704 IS BOTTOM CBP NEXT?
S705 ACQUIRE INFORMATION OTHER THAN CBP
S706 DOES BOTTOM CBP PATTERN MATCH?
S707 SET FIXED VALUE AS BOTTOM CBP
S708 ENTROPY DECODE BOTTOM CBP
S709 IS ACQUISITION OF BLOCK INFORMATION COMPLETE?
S710 DECODE BLOCK IMAGE SIGNAL

(57) Abstract: Involving a step for acquiring a bit stream that includes a flag indicating the segmentation information for block units; a step for decoding and acquiring a top flag that is attached to a top block belonging to a hierarchy with a smaller depth than the depth (N) of a hierarchy and that indicates the quadtree segmentation information for the aforementioned top block; a step for decoding and acquiring a first bottom flag that is attached to a first bottom block from among bottom blocks belonging to the aforementioned hierarchy depth (N) and that indicates the quadtree segmentation information for the aforementioned first bottom block; a step for setting a predetermined first value as a second bottom flag that indicates the quadtree segmentation information for a second bottom block belonging to the aforementioned hierarchy depth (N) in cases in which the already acquired aforementioned top flag and the aforementioned plurality of first bottom flags are a specific combination, and for decoding and acquiring the aforementioned second flag in cases in which the previous are not the aforementioned specific combination; and a step for generating the decoded image signal for the aforementioned blocks in accordance with the aforementioned top flag and first and second bottom flags.

(57) 要約: ブロック単位での分割情報を示すフラグを含むビットストリームを取得するステップと、前記階層の深さNよりも深さが小さい階層に属する上位ブロックに付与された、前記上位ブロックの四分木の分割情報を示す上位フラグを復号し取得するステップと、前記階層の深さNに属する下位ブロックのうち、第1の下位ブロックに付与された、前記上位ブロックの四分木の分割情報を示す第1の下位フラグを復号し取得するステップと、既に取得している前記上位フラグ及び複数の前記第1の下位フラグが特定の組み合わせである場合に、前記階層の深さNに属する第2の下位ブロックの四分木の分割情報を示す第2の下位フラグとして予め定めた第1の値を設定し、前記特定の組み合わせではない場合に、前記第2の下位フラグを復号し取得するステップと、前記上位フラグと、第1及び第2の下位フラグに従って、前記ブロックの復号画像信号を生成するステップと、を具備する。

WO 2011/121843 A1



NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI 添付公開書類:

(BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, — 国際調査報告 (条約第 21 条(3))
NE, SN, TD, TG).

明 細 書

発明の名称： 画像符号化及び画像復号化方法

技術分野

[0001] 本実施形態は、符号化及び復号化方法に関する。

背景技術

[0002] 動画像の符号化技術の1つとしてH. 264（例えば非特許文献1）が知られている。H. 264は復号化（デコード）の方法を定めた規格である。H. 264では、フレーム内およびフレーム間予測の残差（予測誤差）は予め定められた直交変換により変換し、その変換係数を量子化したうえで符号化する。直交変換のサイズは符号化モードにより4×4画素ブロック、8×8画素ブロックの2種類のいずれかが使用される。直交変換の係数は、量子化によりゼロになることが多い。そこでH. 264では、符号量を減らすために、すべての変換係数がゼロであるかを表す1ビットのフラグを符号化し、すべての変換係数がゼロである場合には変換係数の符号化をスキップする。このフラグは、0がすべての係数がゼロであることを表し、1が少なくとも1つの非ゼロの係数があることを表す。

[0003] H. 264では、4×4画素ブロック内の変換数に対するフラグをCBF（coded block flag）と呼び、8×8画素以上のブロックに対するフラグについてはCBP（coded block pattern）と呼ぶ。CBFについては、符号化モードにより、すべての変換係数がゼロであるか否かを表す場合と、変換係数の最初の変換係数（DC成分）以外の係数がすべてゼロであるか否かを表す場合がある。また、4つの4×4画素ブロックは8×8画素のブロックが分割されたものとして扱われ、4つの4×4画素ブロックの変換係数がすべて0である場合には、8×8画素のブロックに対するフラグ（CBP）を0とする。また、16×16画素のブロックに対してもCBPは符号化され、このCBPは、4つの8×8画素の輝度信号（Luminance）ブロック、その16×16画素のブロックに対応する2つの色差信号（Chrominance）ブロック

のすべての変換係数がゼロであるか否かを表す。これらのことからわかるように、変換係数が0であるか否かという情報（CBPおよびCBF）は階層的に符号化されている。

先行技術文献

非特許文献

- [0004] 非特許文献1：ITU-T (Telecommunication Standardization Sector of International Telecommunication Union) Recommendation H.264, “Advanced video coding for generic audiovisual services” ’, 2003.

発明の概要

発明が解決しようとする課題

- [0005] 符号化モードによっては、CBPおよびCBFの符号化において、CBPが0であるなら必ずCBFが0であるというモードが存在する。H. 264では、これらの符号化モードにおいても、CBPが1、すべてのCBFが0という情報を符号化できる。このようなCBPとCBFの組み合わせに符号を割り当てることは冗長である。この冗長性を排除すれば、圧縮率を向上できる。
- [0006] なお、H. 264におけるCBFとCBPの符号化でなくとも、CBPを階層的に符号化する場合には同様の冗長性が生じる。また、H. 264のインター予測ではスキップ予測であるか否かをあらわすモード情報を1ビットで符号化する。今後のH. 264の拡張を考えると、このスキップフラグを階層的に符号化する場合についても同様の冗長性が生じると考えられる。階層的なスキップフラグの符号化における冗長性については、H. 264におけるCBPおよびCBFの冗長性に対する本発明の解決手段と同じ手段で解決できる。
- [0007] 本発明の目的は、階層的なCBPやCBF、スキップフラグの符号化時の冗長性を排除した圧縮データを得ること、及び圧縮データを復号化することにある。

課題を解決するための手段

- [0008] 実施形態の画像復号化方法は、入力画像信号を、階層の深さNに応じたブロックサイズに分割されたブロックを復号する動画像復号化方法において、ブロック単位での分割情報を示すフラグを含むビットストリームを取得するステップと、前記階層の深さNよりも深さが小さい階層に属する上位ブロックに付与された、前記上位ブロックの四分木の分割情報を示す上位フラグを復号し取得するステップと、前記階層の深さNに属する下位ブロックのうち、第1の下位ブロックに付与された、前記上位ブロックの四分木の分割情報を示す第1の下位フラグを復号し取得するステップと、既に取得している前記上位フラグ及び複数の前記第1の下位フラグが特定の組み合わせである場合に、前記階層の深さNに属する第2の下位ブロックの四分木の分割情報を示す第2の下位フラグとして予め定めた第1の値を設定し、前記特定の組み合わせではない場合に、前記第2の下位フラグを復号し取得するステップと、前記上位フラグと、第1及び第2の下位フラグに従って、前記ブロックの復号画像信号を生成するステップと、を具備する。
- [0009] また、実施形態の画像復号化方法は、入力画像信号を、階層の深さNに応じたブロックサイズに分割されたブロックに符号化処理を行う動画像符号化方法において、ブロック単位での分割情報を示すフラグを符号化するステップと、前記階層の深さNよりも深さが小さい階層に属する上位ブロックに付与された、前記上位ブロックの四分木の分割情報を示す上位フラグを取得するステップと、前記階層の深さNに属する下位ブロックのうち、第1の下位ブロックに付与された、前記上位ブロックの四分木の分割情報を示す第1の下位フラグを取得するステップと、既に取得している前記上位フラグ及び複数の前記第1の下位フラグが特定の組み合わせである場合に、前記階層の深さNに属する第2の下位ブロックの四分木の分割情報を示す第2の下位フラグとして予め定めた第1の値を設定し、前記特定の組み合わせではない場合に、前記第2の下位フラグを取得するステップと、前記上位フラグと、第1及び第2の下位フラグに従って、前記ブロックの局部復号画像信号を生成す

るステップと、を具備する。

発明の効果

[0010] 本発明によれば、階層的なCBPやCBF、スキップフラグの符号化時の冗長性を排除した符号化手段により圧縮されたデータが与えられたときに、復号化する画像復号化方法を提供することができる。

図面の簡単な説明

[0011] [図1]符号化又は復号化の対象となる画素ブロックと符号化処理方向を表す図。

[図2]本実施の形態の階層的CBP（下位が4ビットの場合）を示す図。

[図3]図2に対応する単純な階層的CBPを示す図。

[図4]本実施の形態の階層的CBP（下位が2ビットの場合）を示す図。

[図5]本実施の形態のブロック単位のCBPの符号化の動作の一例を示すフローチャート。

[図6]図5の動作を実行する符号化装置のブロック図。

[図7]本実施の形態のブロック単位のCBPの復号化の動作の一例を示すフローチャート。

[図8]図7の動作を実行する符号化装置のブロック図。

[図9]本実施の形態の階層的CBPの復号化方法の一例を示す図。

[図10]図9の別の一例を示す図。

[図11]図9の条件分岐を反転させた復号化方法の一例を示す図。

[図12]8×8画素のブロックに対応するCBPが非0の場合に、4×4画素ブロックのCBFを復号化方法の一例を示す図。

[図13]上位ブロックのサイズが64×64画素、下位ブロックのサイズが32×32画素であり、かつ、下位が4ビットの階層的CBPを復号化方法の一例を示す図。

[図14]図13の別の一例を示す図。

[図15]16×16の画素ブロックに対する1ビットの輝度CBPが与えられた場合に、4×4、8×8、16×8、8×16、16×16の変換につい

ての c b p を得るシンタックスの一例を示す図。

[図16] 単純な階層的スキップフラグを示す図。

[図17] 図 1 6 に対応する本実施の形態の階層的スキップフラグを示す図。

[図18] 本実施の形態のブロック単位のスキップフラグの復号化の動作の一例を示す図。

[図19] 図 1 8 の動作を実現するシンタックスの一例を示す図。

[図20A] 四分木分割に基づいた再帰的ブロック分割の一例を示す図。

[図20B] 四分木分割に基づいた再帰的ブロック分割の一例を示す図。

[図20C] 四分木分割に基づいた再帰的ブロック分割の一例を示す図。

[図21] 四分木分割に基づいた再帰的ブロック分割におけるコーデットブロックフラグのシンタックス構造の一例を示す図。

[図22] 四分木分割に基づいた再帰的ブロック分割におけるコーデットブロックフラグの冗長度を削減するシンタックス構造の一例を示す図。

[図23A] 四分木分割に基づいた再帰的ブロック分割におけるコーデットブロックフラグの冗長度を削減するシンタックス構造の一例を示す図。

[図23B] 四分木分割に基づいた再帰的ブロック分割におけるコーデットブロックフラグの冗長度を削減するシンタックス構造の一例を示す図。

[図24A] 四分木分割に基づいた再帰的ブロック分割における上位コーデットブロックフラグと下位コーデットブロックフラグの符号長の一例を示す図。

[図24B] 四分木分割に基づいた再帰的ブロック分割における上位コーデットブロックフラグと下位コーデットブロックフラグの符号長の一例を示す図。

[図24C] 四分木分割に基づいた再帰的ブロック分割における上位コーデットブロックフラグと下位コーデットブロックフラグの符号長の一例を示す図。

[図24D] 四分木分割に基づいた再帰的ブロック分割における上位コーデットブロックフラグと下位コーデットブロックフラグの符号長の一例を示す図。

[図25] 四分木分割に基づいた再帰的ブロック分割におけるスキップフラグの冗長度を削減するシンタックス構造の一例を示す図。

[図26A] 四分木分割に基づいた再帰的ブロック分割におけるスキップフラグの

冗長度を削減するシンタクス構造の一例を示す図。

[図26B]四分木分割に基づいた再帰的ブロック分割におけるスキップフラグの冗長度を削減するシンタクス構造の一例を示す図。

[図27]コーデットブロックフラグのシンタクス構造とスキップフラグのシンタクス構造とを組み合わせたシンタクス構造の一例を示す図。

[図28]本実施の形態の階層的なスキップフラグの符号化の動作の一例を示すフローチャート。

発明を実施するための形態

[0012] 以下、図面を参照しながら本発明の実施の形態に係る画像復号化方法、画像符号化方法、および装置について詳細に説明する。なお、以下の実施の形態では、同一の番号を付した部分については同様の動作を行うものとして、重ねての説明を省略する。

まず、H. 264で復号化可能なデータを生成する符号化手段について簡単に説明しておく。この符号化手段は、例えば、図1に示すような、フレーム内予測（Intra予測）またはフレーム間予測（Inter予測）、離散コサイン変換（DCT；Discrete Cosine Transform）の近似である整数精度の直交変換およびその逆変換、係数の量子化および逆量子化、コンテキスト適応2値算術符号（CABAC；Context-based Adaptive Binary Arithmetic Code）あるいは可変長符号化（CAVLC；Context-adaptive Variable Length Coding）と呼ばれるエントロピー符号を組み合わせた手段により、H. 264で復号化可能なデータを生成できる。

[0013] 本実施の形態では、符号化においては上位のCBPが1である場合に、下位のCBP（4×4画素の場合はCBF）がすべて0であるという冗長な組み合わせに符号を割り当てないことを考える。本実施の形態の画像復号化方法はそのような圧縮データの復号化を目的とする。また、上位のスキップフラグが0である場合に、下位のスキップフラグがすべて1となるという冗長な組み合わせへの符号の割り当てを避けた符号化を行うものとする。本実施の形態の画像復号化方法はそのような圧縮データの復号化を行う。

[0014] (階層的CBPの冗長性削減、エンコーダ)

以下、残差信号の有無を表す情報をコーデットパターンデータと呼ぶこととする。H. 264の規格書においては、コーデットパターンデータのうち、複数の8 x 8画素あるいは16 x 16画素ブロックに対するフラグをまとめたデータがCBP、1つの4 x 4画素ブロックに対するフラグがCBFに対応する。コーデットパターンデータのうち、上位の1ブロックに対応するコーデットパターンデータをupper_flag、その上位の1ブロックに属するi番目の下位のコーデットパターンデータをlower_flags[i]と呼ぶことにする。例えば、上記非特許文献1において、Luminance信号に対する8 x 8画素ブロックのCBPおよび4 x 4画素ブロックのCBFの場合、upper_flagおよびlower_flags[i]は、それぞれCodedBlockPatternLuma内の1ビット、およびi番目の4 x 4画素ブロックの残差信号に対するcoded_block_flagに対応する。

[0015] MPEG-1/2/4の圧縮ビデオ復号規格やH. 264など多くの規格に定められた復号化の過程では、画像をブロック単位で区切ったうえで、ブロック単位での復号化処理を行う。そこで、以下の説明では、ブロック単位での符号化（エンコード）の過程の一例、およびブロック単位での復号化（デコード）の過程の一例について説明する。画面全体の処理は、ブロック単位の処理を区切られたブロックの数だけ繰り返すことで行える。

[0016] 本発明の実施の形態では、コーデットパターンデータを階層的に符号化する際に、特定のパターンに対応する符号量を減らすことで圧縮率を向上させる手段を提供し、およびそのような手段で生成されたビットストリームを復号化する。特定のパターンの具体的な例（下位が4ビットの場合）を図2に示し、図2に対応した単純な階層的コーデットパターンデータの例（下位が4ビットの場合）を図3に示す（なお、図3は上記非特許文献1のH. 264で使われている）。

[0017] 図3では、下位コーデットパターンデータ（lower_flags）が

0001の場合に、符号化データは0001になる。これに対し、図2の本発明の実施の形態では、下位コードットパターンデータ（lower flags）が0001の場合には最後の1ビットを符号せず、符号化データを000とする。復号化においては、上位コードットパターンデータが1であり、かつ、4ビットの下位コードットパターンのうち、最後の1ビットを除いた3ビットが000であることがわかれば、最後の1ビットは1であることがわかる。すなわち、符号化において1ビット分符号化する必要、および、符号化されたデータの復号化において1ビット分復号化する必要がなくなる場合がある。

[0018] なお、図2では下位のコードットパターンデータを4ビットとしているが、特に4ビットである必要はなく、例えば図4に示す2ビットであってもよい。2ビットの場合には、上位コードットパターンデータが1であり、かつ、2ビットの下位コードットパターンのうち、最初の1ビットが0であることがわかれば、最後の1ビットは1であることがわかる。したがって、この場合も、符号化において1ビット分符号化する必要、および、符号化されたデータの復号化において1ビット分復号化する必要がなくなる場合がある。

[0019] 次に、ブロック単位のCBPの符号化の過程を示すフローチャートを図5、符号化に利用する装置の一例を図6に示す。コードットパターンデータの符号化は、例えば、次の方法により行える。なお、以下の説明では、コードットパターンデータ以外の部分の符号化であるステップ3、ステップ11も含めて説明する。

[0020] ステップ1.（S501）画像入力部601が画像ブロックを読み込む。ここで読み込むブロックは、典型的には、ステップ5における上位コードットパターンデータに対応したブロックと同じか、それより大きなサイズ（例えば、16×16画素）を持つものとする。

[0021] ステップ2.（S502）演算ユニット602がS501で読みこんだブロックに対して符号化すべき予測モードをプログラム用メモリ605から読み込む。なお、この予測モードは予め決めておく場合もあるし、複数の予測

モードのそれぞれでステップS 5 0 3以降を行い最も良いモードを選択する場合もある。他にも予測モードを設定する変形例はたくさんある。

[0022] ステップ3. 演算ユニット6 0 2は予測モードを符号化して、符号化データ出力部6 0 4はビットストリームに出力する。

[0023] ステップ4. (S 5 0 3) 演算ユニット6 0 2は予測モードとすでに復号可能となったデータを利用してブロック予測を行い、入力画像ブロックと予測画像ブロックの差である予測残差を求める。

[0024] ステップ5. (S 5 0 4) 演算ユニット6 0 2は、予測残差にブロック変換（例えば、あらかじめトレーニングデータにより生成したK L Tの基底や、D C Tの基底を利用した直交変換）および量子化を施す。ここで変換を行うブロックは、ステップ1のブロックよりも小さなサイズ（例えば、4 × 4画素）を持つものとする。次に、演算ユニット6 0 2は量子化された変換係数の中に非0の係数が存在するか否かをあらかず情報をブロック単位で算出する。典型的には、この情報は、ブロック内に非0の係数が存在する場合には1、ブロック内の係数がすべて0の場合には0という、ブロックごとに1ビットのフラグ情報になる。この情報を下位のコーデットパターンデータ（下位C B PやC B F）と呼ぶ。次に、下位のコーデットパターンデータの集合（例えば、4つの下位のコーデットパターンデータ）がすべて0か否かという情報を表す、より大きなブロックに対するコーデットパターンデータである上位のコーデットパターンデータ（上位C B P）を求める。典型的には、上位のコーデットパターンデータは、下位のコーデットパターンデータがすべて0であれば0、下位のコーデットパターンデータに1つでも非0の情報があれば1という、ブロックごとに1ビットのフラグ情報になる。ここで、上位コーデットパターンデータに対応するブロックのサイズは、下位コーデットパターンデータよりも大きなサイズ（例えば8 × 8画素）を持つものとする。

[0025] ステップ6. (S 5 0 5) 演算ユニット6 0 2が上位のコーデットパターンデータを符号化して、符号化データ出力部6 0 4がビットストリームに出

力する。

- [0026] ステップ7. (S 5 0 6) 演算ユニット6 0 2は、上位のコーデットパターンデータが0である場合にはステップ8～10を飛ばし、そうでないのならステップ8を実行する。
- [0027] ステップ8. (S 5 0 7) ステップ8～10は下位のコーデットパターンデータを1ビットずつ符号化していく過程である。このステップでは次の条件分岐を実行する。演算ユニット6 0 2は、すでに符号化された下位のコーデットパターンデータが特定のパターンデータである場合（典型的には、最後の1ビットの下位のコーデットパターンデータを符号化する場合であって、かつ、符号化済みの下位のコーデットパターンデータがすべて0である場合）には、次のビットストリームに出力するステップ9を飛ばし、そうでなければステップ9を実行する。
- [0028] ステップ9. (S 5 0 8) 次に、演算ユニット6 0 2は符号化すべき下位のコーデットパターンデータを符号化して、符号化データ出力部6 0 4はビットストリームに出力する。
- [0029] ステップ10. (S 5 0 9) 演算ユニット6 0 2は、下位のコーデットパターンデータをすべて符号化していればステップ11に進め、そうでなければステップ8に戻る。S 5 0 7からS 5 0 9のループはビット数に応じて回る回数が決まり、例えば4ビットであれば4回だけ回すことになる。
- [0030] ステップ11. 符号化データ出力部6 0 4は、コーデットパターンデータが非0のブロックについて、量子化された変換係数を符号化してビットストリームに出力する。
- [0031] なお、ステップ3はS 5 0 1やS 5 0 2より前にあってもよい。エンコーダは、例えば、符号化すべき予測モードを切り替えながらステップ2～11を繰り返し試したうえで、符号化効率が良い予測モードを選択することで、圧縮率を向上させることができる。ただし、デコード可能なビットストリームを出力するために、このような繰り返しは必須ではなく、別の基準で予測モードを決めてもよい。

[0032] 次に、符号化装置について図6を参照して説明する。

本実施の形態の符号化装置は、画像入力部601、演算ユニット602、データ用メモリ603、符号化データ出力部604、プログラム用メモリ605を含む。

[0033] 画像入力部601は画像ブロックを読み込む。演算ユニット602は、画像入力部601が読み込んだ画像ブロックに対し、データ用メモリ603およびプログラム用メモリ605を参照して符号化を行う。データ用メモリ603は、一時記憶装置であり、例えば現在処理しているフレームの前フレームを一時的に記憶する。データ用メモリ603は例えばRAMである。プログラム用メモリ605は、符号化のためのプログラムを記憶している。プログラム用メモリ605は例えばROMまたはRAMである。符号化データ出力部604は、符号化されたデータをビットストリームに出力する。

[0034] (階層的CBPの冗長性削減、デコーダ)

以上の方法で生成されたビットストリームを復号化(ブロック単位のCBPの復号化)する方法のフローチャートの一例を図7に、復号化を実行するための装置の一例を図8に示す。コーデットパターンデータの復号化は、例えば、次のステップ1~10の方法により行える。

[0035] ステップ1. (S701) 符号化データ入力部801が入力データとしてビットストリームを入力する。ビットストリームは、ブロック単位での予測情報、ブロック単位での予測残差信号の有無を表すコーデットパターンデータ、およびブロック単位での予測残差信号をエントロピー符号化した画像データを含む。

[0036] ステップ2. (S702) 次に演算ユニット802が、取得すべきデータが上位のコーデットパターンデータ(例えば、上位のCBP)であるかどうかを判定し、上位のコーデットパターンデータであると判定された場合にはステップ3へ、そうでなければステップ4へ飛ぶ。

[0037] ステップ3. (S703) 演算ユニット802が上位のコーデットパターンデータをエントロピー復号により取得する。ステップ9へ飛ぶ。

- [0038] ステップ4. (S704) 次に演算ユニット802が、取得すべきデータが上位コーデットパターンデータに属するコーデットパターンデータである下位のコーデットパターンデータ（例えば、下位のCBPあるいは下位のCBF）であるかどうかを判定し、下位のコーデットパターンデータであると判定された場合にはステップ6へ、そうでなければステップ5へ飛ぶ。
- [0039] ステップ5. (S705) 次に演算ユニット802が取得すべき（コーデットパターンデータ以外の）データを取得しステップ9へ飛ぶ。
- [0040] ステップ6. (S706) 演算ユニット802が、すでに取得された下位のコーデットパターンデータが予め定められたパターンであるかどうかを判定し、予め定められたパターンである場合にはステップ7へ飛び、そうでなければステップ8へ飛ぶ。例えば、図2の例においては、予め定められたパターンとは、下位のコーデットパターンデータが4ビットであり、かつ、すでに下位のコーデットパターンデータが3ビット取得されていて次に取得すべきデータが4ビット目のコーデットパターンデータであり、かつ、すでに取得された3ビットが000であるパターンを指す。
- [0041] ステップ7. (S707) 次に演算ユニット802が、取得すべき下位のコーデットパターンデータとして、予め定められた固定値を設定し、設定後ステップ9へ飛ぶ。例えば、図2の例では、次に取得すべき4ビット目のコーデットパターンデータとして1を設定する。
- [0042] ステップ8. (S708) 次に演算ユニット802が、取得すべき下位のコーデットパターンデータをエンтроピー復号により取得する。
- [0043] ステップ9. (S709) 演算ユニット802がブロック情報の取得が完了しているかどうかを判定し、取得が完了していればステップ10へ飛び、そうでなければステップ2へ飛ぶ。ブロック情報は、復号された、ブロック単位での予測情報およびブロック単位での予測残差信号を示す。すなわち、このフローでは明示されていないが、ビットストリームに含まれる情報がブロック単位での予測情報である場合にはビットストリームを復号化しブロック単位での予測情報を得るステップがあり、ビットストリームに含まれる情

報がブロック単位での予測残差信号である場合にはビットストリームを復号化しブロック単位での予測残差信号を得るステップがある。

[0044] ステップ10. (S710) 演算ユニット802が取得されたブロック情報に基づきブロック画像信号を復元し、画像出力部804が復元された画像を出力する。

[0045] 次に、復号化装置について図8を参照して説明する。

本実施の形態の復号化装置は、符号化データ入力部801、演算ユニット802、データ用メモリ803、画像出力部804、プログラム用メモリ805を含む。

[0046] 符号化データ入力部801は入力データとしてビットストリームを入力する。演算ユニット802は、符号化データ入力部801が入力したビットストリームに含まれるブロックを、ブロックの切れ目を判断しながら、データ用メモリ803およびプログラム用メモリ805を参照して復号化を行う。データ用メモリ803は、一時記憶装置であり、例えば符号化データ入力部801が入力したビットストリームを一時的に記憶する。プログラム用メモリ805は、復号化のためのプログラムを記憶していて、例えば下記図9または図10に示す擬似プログラムに対応するプログラムを格納している。

[0047] 次に、下位のコーデットパターンデータが4ビットの階層的CBPの場合の復号化方法のシンタックスの一例を図9に、下位のコーデットパターンデータがnum_lower_blocksビットの階層的CBPの場合の復号化方法のシンタックスの一例を図10に示す。図9、図10において、ae(v)とされた行はエントロピー復号(例えばCABAC)により1ビットの情報を取得することを表し、そうでない行はその行を実行することを表す。「(upper_flag is already decoded)」は、上位のコーデットパターンデータがすでにupper_flagとして取得されていることを表す。「(other decoding processes, optional)」は、もし他のシンタックス処理が必要であれば、そのシンタックス処理を実行することを表す。なお、コーデッ

トパターンデータが特定の組み合わせであるかを判定する条件分岐判定は反転させることが可能である。図9の条件分岐判定を反転させた例を図11に示す。他の図についても同様である。

[0048] (CBPとCBF)

H. 264では、4 x 4画素のブロックに対応したコーデットパターンデータである`coded_block_flag`は、第1の`coded_block_flag`、第1の`coded_block_flag`が非0の場合には第1の変換係数、第2の`coded_block_flag`、第2の`coded_block_flag`が非0の場合には第2の変換係数、というように、変換係数とあわせて復号化される。この場合、上位のコーデットパターンデータとして8 x 8画素のブロックに対応するCBP、下位のコーデットパターンデータとして8 x 8画素のブロックを構成する各4 x 4画素のブロックに対応したCBFを考えると、シンタックスは図12のように表せる。図12は、8 x 8画素のブロックに対応するCBPが非0の場合に、4 x 4画素ブロックのCBFを復号化する方法の一例を示す。

[0049] 図12において、引数`blockIndex`は、処理中の4 x 4画素ブロックがそれを含む8 x 8画素のブロック内で先頭のブロックを0としたときのインデックスを表し、(`decode_residual_coefficients_coeffLevel[i]`)は変換係数を復号化することを表す。また、最初の4 x 4画素ブロックを復号化する前に、`nonzero_coded_block_flag_found`には0を設定しておくものとする。8 x 8画素のブロックを構成する4 x 4画素ブロックは4つあるから、`blockIndex`が3であり、かつ、他の`coded_block_flag`が0である場合には、`coded_block_flag`を復号化せずに1とすることができる。そうでない場合には、`coded_block_flag`を復号化する必要がある。なお、図12には含まれていないが、予測モードによっては常に`coded_block_flag`を復号化する必要があるため、図12の関数を呼び出す前に、常に`coded__`

`block_flag`を復号化する必要があるかを、判定しておく必要がある。常に`coded_block_flag`を復号化する必要がある予測モードとしては、例えば、H. 264のIntra 16x16予測があげられる。このモードでは、輝度ブロックに対応したCBPの表現として、16x16画素ブロックを構成する4つの8x8ブロックに対する4ビットのCBPについて、すべてのビットが0であるかすべてのビットが1であるかの2通りしか用意されていない。この場合、上位のCBPが1かつ下位のCBFがすべて0という符号を用意しておかなければ、16x16画素内の任意の8x8画素ブロックにおいて、8x8画素ブロック内のCBFがすべて0になるというコードットパターンデータを表現できない。したがって、H. 264のIntra 16x16予測では、常に`coded_block_flag`を復号化する必要がある。

[0050] (上位CBPと下位CBP)

H. 264では、上位のコードットパターンデータを持つブロック（マクロブロック）のサイズは16x16画素である。しかし、符号化効率を考えると、例えばマクロブロックのサイズを32x32画素や64x64画素に拡張するという拡張も考えられる。これを拡張マクロブロックと呼ぶことにする。拡張マクロブロックでは、例えば、32x32画素の拡張マクロブロックを構成する16x16画素のブロックは、輝度ブロックのみを考えれば4個、輝度ブロック4つに加えてCb、Crの色差ブロックまで考えれば12個存在することになる。CBPは0となることが多く、0のフラグを12個も符号化することは避けたい。そこでマクロブロックの拡張として、32x32画素の拡張マクロブロックを構成するCBPがすべて0であるか、1つ以上の非0のビットを持つCBPであるかというコードットパターンデータを符号化することが考えられる。この実施例では、32x32画素の拡張マクロブロックに対するコードットパターンデータを`coded_block_pattern_32`と呼ぶことにする。`coded_block_pattern_32`は1ビットのフラグ情報である。さらに大きな拡張マク

ロブロックとして、 64×64 画素のレベル2拡張マクロブロックを考えるなら、レベル2拡張マクロブロックを構成する 32×32 画素のCBPである`coded_block_pattern_32`がすべて0であるか、1つ以上の非0のビットを持つCBPであるかを符号化することが考えられる。この実施例では、 64×64 画素の拡張マクロブロックに対するコードットパターンデータを`coded_block_pattern_64`と呼ぶこととし、その 64×64 画素ブロックを構成する4つの 32×32 画素ブロックについては、インデックスを付与して`coded_block_pattern_32[0] ~ coded_block_pattern_32[3]`と呼ぶことにする。

[0051] `coded_block_pattern_64`および`coded_block_pattern_32`をデコードする処理の流れの一例を図13、別の一例を図14に示す。双方の図は、上位ブロックのサイズが 64×64 画素、下位ブロックのサイズが 32×32 画素であり、かつ、下位が4ビットの階層的CBPを復号化する方法の一例を示す。

[0052] 「(other decoding processes, optional)」は、もし他のシンタックス処理が必要であれば、そのシンタックス処理を実行することを表す。図13は、`coded_block_pattern_64`と`coded_block_pattern_32`が符号化されたデータを復号化する場合の一例を表している。図13では、まず`coded_block_pattern_64`を復号化し、それが0でない場合には`coded_block_pattern_32`の各要素を復号化している。この際、`coded_block_pattern_32[3]`については、`coded_block_pattern_32[0] ~ coded_block_pattern_32[2]`のうち1つ以上が非0であれば復号化するが、`coded_block_pattern_32[0] ~ coded_block_pattern_32[2]`のすべてが0である場合には`coded_block_pattern_32[3]`を1とす

る。図14は、図13における`coded_block_pattern_64`と`coded_block_pattern_32`が別の関数として表現された場合の一例を表している。図14において、`macroblock__cluster__residual__64x64()`は 64×64 画素のレベル2拡張マクロブロックの残差を復号化するための関数であり、`macroblock__cluster__residual__32x32()`は 32×32 画素の拡張マクロブロックの残差を復号化するための関数である。図13に対し、図14では、`coded_block_pattern_64`の復号化が`macroblock__cluster__residual__64x64()`で行われ、`coded_block_pattern_32`の各要素の復号化は`macroblock__cluster__residual__32x32()`で行われている点が異なる。`coded_block_pattern_32[0] ~ coded_block_pattern_32[2]`のすべてが0である場合に、`coded_block_pattern_32[3]`を復号化せず固定値1とする点は図13と同じである。

[0053] (階層的CBPの具体例)

シンタックスの一例として、 16×16 画素のブロックに対して1ビットの輝度CBPが符号化および復号化され、輝度CBPが1の場合にのみ、 16×16 画素のブロック内に属する各ブロックについて、変換係数の有無をCBPとして符号化および復号化される例を、図15のシンタックスにしたがって説明する。図15は、 16×16 の画素ブロックに対する1ビットの輝度CBPが与えられた場合に、 4×4 、 8×8 、 16×8 、 8×16 、 16×16 の変換についての`c_b_p`を得るシンタックスの一例を示す。

[0054] 変換のサイズとして、 4×4 、 8×8 、 16×8 、 8×16 、 16×16 の5通りを考える。変換タイプ「`cur__transform__type`」は、 4×4 に対応する0、 8×8 に対応する1、 $16 \times 8 / 8 \times 16 / 16 \times 16$ のいずれかに対応する2の3値を取るものとする。「`cur__tra`

「`nsform__type`」が2の場合、変換サイズは、予測ブロックのサイズが 16×8 であれば 16×8 、 8×16 であれば 8×16 、それ以外であれば 16×16 とする。変換サイズが 4×4 および 8×8 の場合には、4つの 8×8 ブロックに対する4ビットのCBPが必要である。変換サイズが 16×8 および 8×16 の場合には、それぞれ、2つの 16×8 ブロックおよび2つの 8×16 ブロックに対する2ビットのCBPが必要である。変換サイズが 16×16 の場合には、1ビットのCBPが必要である。

[0055] この場合、図15に示すように、`cur__transform__type`が0または1の場合には図2に対応した読み込みを行い、`cur__transform__type`が2かつ変換サイズが $16 \times 8 / 8 \times 16$ の場合には図4に対応した読み込みを行い、`cur__transform__type`が2かつ変換サイズが 16×16 の場合には与えられたCBPをそのまま利用することで、本発明を利用できることになる。なお、図15において、「`cbp__luma__1bit`」は1ビットの輝度CBPを表し、「`MbPartWidth (mb__type)`」および「`MbPartHeight (mb__type)`」は予測ブロックのサイズを表し、`cbp`は 16×16 輝度ブロックを4つの 8×8 ブロックに分割した場合の、各 8×8 輝度ブロックに対応した4ビットの`cbp`を表す。図15のシンタックスでは、 $16 \times 8 / 8 \times 16 / 16 \times 16$ の場合について、`cbp`を 8×8 ブロックにあわせて4ビットとしているが、これらの場合に`cbp`を 8×8 ブロックにあわせる必要はなく、それぞれ2ビット、2ビット、1ビットの`cbp`として読み込んでもよい。

[0056] (四分木分割による再帰構造をもった階層ブロックにおけるCBF符号化)

本実施形態の変更例として、H. 264のブロック構造を $M \times M$ サイズ（ M は2のべき乗）のブロックに階層的に分割することを考える。この際、上位ブロックサイズを $2N \times 2N$ 、下位ブロックサイズを $N \times N$ （ N は2のべき乗）で表現する。図20A、図20Bに $2N \times 2N$ サイズと $N \times N$ サイズ

の符号化ブロックを示す。上位ブロックサイズを分割する際には、分割を示す1ビットのフラグを用いて分割の有無を表現する。CBFについても同様に、上位ブロックに上位CBFが存在し、下記ブロックには0から3で示される4つの $N \times N$ ブロック毎に4つの下位CBFが存在する。ここで上位CBFが1で有りかつ下位CBFが全て0の場合は、上位CBFが0である場合と同じ表現となるため冗長である。このようなCBFの符号化を考えた場合、上位CBFは、上位ブロックを分割するかどうかを示す情報と等価となる。

[0057] $N \times N$ を更に分割する場合においても、上記関係を用いて再帰的に表現することが可能である。例えば、上位ブロックと下位ブロックを再帰的に組み合わせた時の階層的な四分木分割の一例を図20Cに示す。図20Bでは下位ブロックとなっていたブロック2が、図20Cでは更に四分木分割されている。この場合、図20Cのブロック2には上位ブロックとしてのCBFが存在するとともに、ブロック2を四分木分割しているので下位ブロックの4つのCBFも存在することになる。ここでは、ブロック2の下位ブロックの番号を0'、1'、2'、3'として上位ブロックと区別している。

[0058] つまり、上位ブロックと下位ブロックの構成は、階層の深さによって変化する。例えば、階層の深さ (layer) で表現される上位ブロックは、(layer+1)の階層では4つの下位ブロックを持つことになる。同様に (layer+1)の階層で表現される上位ブロックは、(layer+2)の階層では4つの下位ブロックを持つことになる。CBFについても同様の階層化による四分木分割が可能である。

[0059] このような階層的なブロック分割に対応するシンタックスの一例として、階層構造を再帰的に表現したときのCBFの関係を図21に示す。図中に示される `transform_unit()` は、図中太字で表されている、上位ブロックの非0の変換係数の有無を表す `upper_coded_block_flag` と、下位ブロックの非0の変換係数の有無を表す `lower_coded_block_flag` を符号化するシンタックス構造を示してい

る。図中の `currBlockPos` は、当該ブロックの位置を示す情報であり、`layer` は、階層構造の深さを示す。例えば、`currBlockPos 0` は図 20B におけるブロック 0 を示し、`currBlockPos 3` は図 20B におけるブロック 3 を示している。`layer` の一例として、図 20A で $2N \times 2N$ サイズが 64×64 の例を示すと、 $N \times N$ サイズは 32×32 となり、`layer` は 1 となる。更に $N \times N$ を分割する場合、 $N/2 \times N/2$ サイズは 16×16 となり、`layer` は 2 となる。

[0060] `currLowerAvailable` は、当該ブロックが下位 CBF を保持することができるかどうかを示す内部変数である。言い換えれば、当該ブロックが、分割可能かどうかを示している。例えば、予め定めた最小分割ブロックサイズよりも当該ブロックを分割したときのブロックサイズが小さくなる場合、下位 CBF を持つことができないため `currLowerAvailable` は 0 となる。一方、当該ブロックを更に分割することが可能である場合 `currLowerAvailable` は 1 となる。

[0061] `upper__coded__block__flag` は、上位ブロックの非 0 の変換係数の有無を示すフラグであり、言い換えれば上位ブロックを分割するかどうかを表している。分割する場合は 1、分割しない場合は 0 となる。`upper__coded__block__flag` が 1 である場合、上位ブロックに非 0 の変換係数が存在するため、更に下位 CBF を持つことが可能となる。関数 `calculateBlockPos()` は、当該ブロックを四分木分割した際、`i` (`i` は 0 から最大 3 まで) で示される分割後のブロックの位置を返す関数である。`currBlockPos 0` から 3 は、最大 4 までの分割後のブロックの位置を示す。

[0062] `lower__coded__block__flag` は、下位ブロックの非 0 の変換係数の有無を示すシンタクス要素であり、当該下位ブロックに非 0 の変換係数が存在する場合は 1 を、存在しない場合は 0 となる。`codedBlockFlag` は階層構造毎に `lower__coded__block__flag` を保持する内部変数であり、下位ブロックに該当する当該ブロックが

ら `layer` が 0 で示される上位ブロックまでの `lower__coded__block__flag` が保持されている。

[0063] `residual__block()` は、CBF 以外の変換係数に関連する情報を符号化する関数である。なお、本関数は非 0 の変換係数が存在する時だけ (`lower__coded__block__flag` が 1 のときだけ) 呼び出される。更に、本関数は `transform__unit()` 以外の関数で呼び出される構成としてもよい。

[0064] 図 21 で示される再帰構造を持ったシンタクス構造を考えた場合、上位 CBF が 1 である場合に、下位の CBF (4 x 4 画素の場合は CBF) がすべて 0 であるという冗長な組み合わせが存在する。

[0065] (四分木分割による再帰構造をもった階層ブロックにおける CBF の冗長度削減)

図 21 で説明した再帰構造を持つ CBF のシンタクス構造時にも発生する下位 CBF の冗長性を削減したシンタクス構造の一例が図 22 に示されている。図 22 では、`transform__unit()` 関数に、`DecodedSkipFlag` が引数として追加されている。`DecodedSkipFlag` は、当該ブロックにおいて `lower__coded__block__flag` を符号化又は復号するかどうかを示す 1 ビットの内部変数である。上位 CBF が存在した場合 (上位ブロックの `upper__coded__block__flag` が 1 の場合) で且つ、4 つの下位 CBF の内の最初の 3 つの `lower__coded__block__flag` が全て 0 である場合、`DecodedSkipFlag` が 1 となる。そうでない場合は、`DecodedSkipFlag` は 0 に設定される。

[0066] 逆にいえば、`DecodedSkipFlag` が 1 の場合、下位ブロックの最初から 3 番目までの CBF が 0 であるため、4 番目に相当する下位ブロックの CBF (`lower__coded__block__flag`) は自動的に 1 に設定される。ここでは、最初の 3 つの `lower__coded__block__flag` が全て 0 である場合を説明したが、符号化及び復号の際に

最後に処理されるブロック以外が全て0である場合にDecodedSkipFlagが1とすることも容易に実現可能である。このようなシンタクス構造をもつことで、四分木分割されたシンタクス構造における下位CBFの冗長性を削減することが可能となる。

[0067] (シンタクス要素の分割)

なお、本実施の形態に関わるシンタクス要素のupper__coded__block__flagとlower__coded__block__flagを別々の関数として分けることも可能である。この例が図23A及び図23Bに記述されている。

[0068] 図23Aでは、upper__coded__block__flagのみを再帰的に符号化するシンタクス構造である。図中に示される(・・・)は、CBF以外の様々なシンタクス要素、関数が記述されてもよいことを意味している。upper__coded__block__flagが0となる場合に、transform__unit__tree()関数が呼ばれる。一方、図23Bにtransform__unit__tree()関数が示されている。transform__unit__tree()関数は、lower__coded__block__flagのみを符号化する関数である。以上のように上位ブロックと下位ブロックを分割しても、分割しない図22と同一の機能を実現できる。

[0069] (CBF符号化例1：冗長度有り)

図20Bの場合で、4つの下位ブロックと上位ブロックのCBFの符号長を図21のシンタクスに従って算出した場合の符号量は、図3で示されている。下位CBFの組み合わせ(lower__codes)が0001の場合に、符号化データは0001となり、4ビットが必要となる。これに対し、図22のシンタクスに従って算出した場合の符号量は、図2で示されている。下位CBF(lower__codes)が0001の場合には最後の1ビットを符号せず、符号化データを000とする。これにより、冗長な最後の1ビットの符号量を削減できる。

[0070] 次に図20Bの場合に、図21のシンタクス構造を用いてCBFを符号化・復号化した場合の、各ブロックの符号長を図24Aに示す。本図を用いて符号化・復号化の流れを説明する。ここで、図20Bでは、ブロック3のみが非0の変換係数を有し、それ以外のブロックは非0の変換係数を持たない場合を考える。

[0071] 先ず、図20Aの $2N \times 2N$ の上位ブロック0”に対して`transform_unit()`が呼ばれる。ここで上位ブロックは4つの下位ブロックに分割されているため、`upper_coded_block_flag`は1として符号化される。次に図20Bのブロック0に対して`transform_unit()`が呼び出される。ここで $N \times N$ のブロック0は下位ブロックに分割されていないため、`upper_coded_block_flag`は0として符号化・復号化される。更に当該ブロックは非ゼロの変換係数を持たないため、`lower_coded_block_flag`は0として符号化・復号化される。

[0072] 次にブロック1に対して`transform_unit()`が呼び出される。ここで $N \times N$ のブロック1は下位ブロックに分割されていないため、`upper_coded_block_flag`は0として符号化・復号化される。更に当該ブロックは非ゼロの変換係数を持たないため、`lower_coded_block_flag`は0として符号化・復号化される。ブロック2についても同様の符号化・復号化処理が施される。最後にブロック3に対して`transform_unit()`が呼び出される。ここで $N \times N$ のブロック3は下位ブロックに分割されていないため、`upper_coded_block_flag`は0として符号化・復号化される。当該ブロックは非ゼロの変換係数を持つため、`lower_coded_block_flag`は1として符号化・復号化される。

[0073] 以上の処理を経て、符号化・復号化されたCBFの情報は5ビットの`upper_coded_block_flag`と4ビットの`lower_coded_block_flag`となり9ビットの符号長となる。

[0074] (C B F 符号化例 1 : 冗長度削減手法)

次に図 20 B の場合に、図 22 のシンタクス構造を用いて C B F を符号化・復号化した場合の、各ブロックの符号長を図 24 B に示す。本図を用いて符号化・復号化の流れを説明する。ここでは、先ほどと同様に、ブロック 3 のみが非 0 の変換係数を有し、それ以外のブロックは非 0 の変換係数を持たない場合を考える。

[0075] ブロック 0” 及びブロック 0 からブロック 2 までは前述と同一の処理が施される。なお、このとき、`DecodeSkipFlag` は 0 である。最後にブロック 3 に対して `transform_unit()` が呼び出される。ここで、ブロック 0、ブロック 1、ブロック 2 の 3 つの下位ブロックの C B F が全て 0 となるため、`DecodeSkipFlag` が 1 に設定される。ブロック 3 は下位ブロックに分割されていないため、`upper_coded_block_flag` は 0 として符号化・復号化される。`DecodeSkipFlag` が 1 であるため、`lower_coded_block_flag` は符号化・復号化されず、内部的に 1 が設定される。

[0076] 以上の処理を経て、符号化・復号化された C B F の情報は 5 ビットの `upper_coded_block_flag` と 3 ビットの `lower_coded_block_flag` となり 8 ビットの符号長となる。これにより、1 ビットの無駄な符号を削減できる。

[0077] (C B F 符号化例 2 : 冗長度有り)

次に図 20 C の場合に、図 21 のシンタクス構造を用いて C B F を符号化・復号化した場合の、各ブロックの符号長を図 24 C に示す。本図を用いて符号化・復号化の流れを説明する。ここで、図 20 C では、ブロック 3’ のみが非 0 の変換係数を有し、それ以外のブロックは非 0 の変換係数を持たない場合を考える。

[0078] ブロック 0”、ブロック 0 及びブロック 1 は、図 24 A の説明と同一の処理が施される。続いてブロック 2 に対して `transform_unit()` が呼び出される。ここでブロック 2 は 4 つの下位ブロックに分割されてい

るため、`upper__coded__block__flag`は1として符号化・復号化される。続いてブロック2の下位ブロックであるブロック0'に対して`transform__unit()`が呼び出される。ここで $N/2 \times N/2$ のブロック0'は下位ブロックに分割されていないため、`upper__coded__block__flag`は0として符号化・復号化される。更に当該ブロックは非ゼロの変換係数を持たないため、`lower__coded__block__flag`は0として符号化・復号化される。

[0079] 次にブロック1'に対して`transform__unit()`が呼び出される。ここで $N/2 \times N/2$ のブロック1'は下位ブロックに分割されていないため、`upper__coded__block__flag`は0として符号化・復号化される。更に当該ブロックは非ゼロの変換係数を持たないため、`lower__coded__block__flag`は0として符号化・復号化される。ブロック2'についても同様の符号化・復号化処理が施される。続いてブロック3'に対して`transform__unit()`が呼び出される。ここで $N/2 \times N/2$ のブロック3'は下位ブロックに分割されていないため、`upper__coded__block__flag`は0として符号化・復号化される。当該ブロックは非ゼロの変換係数を持つため、`lower__coded__block__flag`は1として符号化・復号化される。

[0080] 引き続き、ブロック3に対して`transform__unit()`が呼び出される。ここで $N \times N$ のブロック3は下位ブロックに分割されていないため、`upper__coded__block__flag`は0として符号化・復号化される。更に当該ブロックは非ゼロの変換係数を持たないため、`lower__coded__block__flag`は0として符号化・復号化される。

[0081] 以上の処理を経て、符号化・復号化されたCBFの情報は9ビットの`upper__coded__block__flag`と7ビットの`lower__coded__block__flag`となり16ビットの符号長となる。

[0082] (CBF符号化例2：冗長度削減手法)

次に図20Cの場合に、図22のシンタクス構造を用いてCBFを符号化・復号化した場合の、各ブロックの符号長を図24Dに示す。本図を用いて符号化・復号化の流れを説明する。ここでは、先ほどと同様に、ブロック3'のみが非0の変換係数を有し、それ以外のブロックは非0の変換係数を持たない場合を考える。

[0083] ブロック0”、ブロック0からブロック1、及びブロック0'からブロック2'までは前述と同一の処理が施される。なお、このとき、DecodeSkipFlagは0である。続いてブロック3'に対してtransform_unit()が呼び出される。ここで、ブロック0'、ブロック1'、ブロック2'の3つの下位ブロックのCBFが全て0となるため、DecodeSkipFlagが1に設定される。ブロック3'は下位ブロックに分割されていないため、upper_coded_block_flagは0として符号化・復号化される。更にDecodeSkipFlagが1であるため、lower_coded_block_flagは符号化・復号化されず、内部的に1が設定される。

[0084] 引き続き、ブロック3に対してtransform_unit()が呼び出される。ここでNxNのブロック3は下位ブロックに分割されていないため、upper_coded_block_flagは0として符号化・復号化される。更に当該ブロックは非ゼロの変換係数を持たないため、lower_coded_block_flagは0として符号化・復号化される。

[0085] 以上の処理を経て、符号化・復号化されたCBFの情報は9ビットのupper_coded_block_flagと6ビットのlower_coded_block_flagとなり15ビットの符号長となる。これにより、1ビットの無駄な符号を削減できる。

[0086] (図22に準じた符号化フローチャート)

図22のシンタクス構造に基づいて、本実施の形態を図5のフローチャート及び図6の符号化装置に沿って説明する。コードットパターンデータの符

号化は、例えば、次の方法により行える。なお、以下の説明では、コードットパターンデータ以外の部分の符号化であるステップ3、ステップ11も含めて説明する。

- [0087] ステップ1. (S501) 画像入力部601が画像ブロックを読み込む。ここで読み込むブロックは、典型的には、ステップ5における上位コードットパターンデータに対応したブロックと同じか、それより大きなサイズ（例えば、 64×64 画素）を持つものとする。
- [0088] ステップ2. (S502) 演算ユニット602がS501で読みこんだブロックに対して符号化すべき予測モードをプログラム用メモリ605から読み込む。なお、この予測モードは予め決めておく場合もあるし、複数の予測モードのそれぞれでステップS503以降を行い最も良いモードを選択する場合もある。他にも予測モードを設定する変形例はたくさんある。
- [0089] ステップ3. 演算ユニット602は予測モードを符号化して、符号化データ出力部604はビットストリームに出力する。
- [0090] ステップ4. (S503) 演算ユニット602は予測モードとすでに復号可能となったデータを利用してブロック予測を行い、入力画像ブロックと予測画像ブロックの差である予測残差を求める。
- [0091] ステップ5. (S504) 演算ユニット602は、予測残差にブロック変換（例えば、あらかじめトレーニングデータにより生成したKLTの基底や、DCTの基底を利用した直交変換）および量子化を施す。ここで変換を行うブロックは、ステップ1のブロックよりも小さなサイズ（例えば、 4×4 画素）を持つものとする。次に、演算ユニット602は量子化された変換係数の中に非0の係数が存在するか否かをあらかず情報をブロック単位で算出する。典型的には、この情報は、ブロック内に非0の係数が存在する場合には1、ブロック内の係数がすべて0の場合には0という、ブロックごとに1ビットのフラグ情報になる。この情報を下位のコードットパターンデータ（下位CBPやCBF）と呼ぶ。次に、下位のコードットパターンデータの集合（例えば、4つの下位のコードットパターンデータ）がすべて0か否かと

いう情報を表す、より大きなブロックに対するコーデットパターンデータである上位のコーデットパターンデータ（上位CBP）を求める。典型的には、上位のコーデットパターンデータは、下位のコーデットパターンデータがすべて0であれば0、下位のコーデットパターンデータに1つでも非0の情報があれば1という、ブロックごとに1ビットのフラグ情報になる。ここで、上位コーデットパターンデータに対応するブロックのサイズは、下位コーデットパターンデータよりも大きなサイズ（例えば8×8画素）を持つものとする。この処理を階層的に行って最下位のCBPから最上位のCBPまでを設定する。例えば最下位のブロックを4×4画素、最上位のブロックを64×64画素とした場合、階層の深さが0（64×64画素ブロック）から5（4×4画素ブロック）までで表現される。

- [0092] ステップ6.（S505）演算ユニット602が上位のコーデットパターンデータを符号化して、符号化データ出力部604がビットストリームに出力する。
- [0093] ステップ7.（S506）演算ユニット602は、上位のコーデットパターンデータが0である場合にはステップ8～10を飛ばし、そうでないのならステップ8を実行する。
- [0094] ステップ8.（S507）ステップ8～10は下位のコーデットパターンデータを1ビットずつ符号化していく過程である。このステップでは次の条件分岐を実行する。演算ユニット602は、すでに符号化された下位のコーデットパターンデータが特定のパターンデータである場合（典型的には、最後の1ビットの下位のコーデットパターンデータを符号化する場合であって、かつ、符号化済みの下位のコーデットパターンデータがすべて0である場合）には、次のビットストリームに出力するステップ9を飛ばし、そうでなければステップ9を実行する。
- [0095] ステップ9.（S508）次に、演算ユニット602は符号化すべき下位のコーデットパターンデータを符号化して、符号化データ出力部604はビットストリームに出力する。

- [0096] ステップ10. (S509) 演算ユニット602は、下位のコーデットパターンデータをすべて符号化していればステップ11に進め、そうでなければステップ8に戻る。S507からS509のループはビット数に応じて回る回数が決まり、例えば4ビットであれば4回だけ回すことになる。また、このステップ7~10は、上位ブロックが分割されなくなるまで（ステップ10の判定がYesになるまで）、階層に応じて再帰的に繰り返される。
- [0097] ステップ11. 符号化データ出力部604は、コーデットパターンデータが非0のブロックについて、量子化された変換係数を符号化してビットストリームに出力する。
- [0098] なお、ステップ3はS501やS502より前にあってもよい。エンコーダは、例えば、符号化すべき予測モードを切り替えながらステップ2~11を繰り返し試したうえで、符号化効率が良い予測モードを選択することで、圧縮率を向上させることができる。ただし、デコード可能なビットストリームを出力するために、このような繰り返しは必須ではなく、別の基準で予測モードを決めてもよい。
- [0099] (図22に準じた復号フローチャート)
- 次に本実施の形態で説明した図22のシンタクス構造を図7のフローチャート及び図8の復号装置に沿って説明する。
- [0100] 以上の方法で生成されたビットストリームを復号化（ブロック単位のCBPの復号化）する方法のフローチャートの一例を図7に、復号化を実行するための装置の一例を図8に示す。コーデットパターンデータの復号化は、例えば、次のステップ1~10の方法により行える。
- [0101] ステップ1. (S701) 符号化データ入力部801が入力データとしてビットストリームを入力する。ビットストリームは、ブロック単位での予測情報、ブロック単位での予測残差信号の有無を表すコーデットパターンデータ、およびブロック単位での予測残差信号をエントロピー符号化した画像データを含む。
- [0102] ステップ2. (S702) 次に演算ユニット802が、取得すべきデータ

が上位のコーデットパターンデータ（例えば、上位のCBP）であるかどうかを判定し、上位のコーデットパターンデータであると判定された場合にはステップ3へ、そうでなければステップ4へ飛ぶ。

[0103] ステップ3. (S703) 演算ユニット802が上位のコーデットパターンデータをエントロピー復号により取得する。ステップ9へ飛ぶ。

[0104] ステップ4. (S704) 次に演算ユニット802が、取得すべきデータが上位コーデットパターンデータに属するコーデットパターンデータである下位のコーデットパターンデータ（例えば、下位のCBPあるいは下位のCBF）であるかどうかを判定し、下位のコーデットパターンデータであると判定された場合にはステップ6へ、そうでなければステップ5へ飛ぶ。

[0105] ステップ5. (S705) 次に演算ユニット802が取得すべき（コーデットパターンデータ以外の）データを取得しステップ9へ飛ぶ。

[0106] ステップ6. (S706) 演算ユニット802が、すでに取得された下位のコーデットパターンデータが予め定められたパターンであるかどうかを判定し、予め定められたパターンである場合にはステップ7へ飛び、そうでなければステップ8へ飛ぶ。例えば、図2の例においては、予め定められたパターンとは、下位のコーデットパターンデータが4ビットであり、かつ、すでに下位のコーデットパターンデータが3ビット取得されていて次に取得すべきデータが4ビット目のコーデットパターンデータであり、かつ、すでに取得された3ビットが000であるパターンを指す。

[0107] ステップ7. (S707) 次に演算ユニット802が、取得すべき下位のコーデットパターンデータとして、予め定められた固定値を設定し、設定後ステップ9へ飛ぶ。例えば、図2の例では、次に取得すべき4ビット目のコーデットパターンデータとして1を設定する。

[0108] ステップ8. (S708) 次に演算ユニット802が、取得すべき下位のコーデットパターンデータをエントロピー復号により取得する。

[0109] ステップ9. (S709) 演算ユニット802がブロック情報の取得が完了しているかどうかを判定し、取得が完了していればステップ10へ飛び、

そうでなければステップ2へ飛ぶ。ブロック情報は、復号された、ブロック単位での予測情報およびブロック単位での予測残差信号を示す。すなわち、このフローでは明示されていないが、ビットストリームに含まれる情報がブロック単位での予測情報である場合にはビットストリームを復号化しブロック単位での予測情報を得るステップがあり、ビットストリームに含まれる情報がブロック単位での予測残差信号である場合にはビットストリームを復号化しブロック単位での予測残差信号を得るステップがある。

[0110] ステップ10. (S710) 演算ユニット802が取得されたブロック情報に基づきブロック画像信号を復元し、画像出力部804が復元された画像を出力する。

[0111] このステップ2～9は、上位ブロックが分割されなくなるまで（ステップ9の判定がYesになるまで）、階層に応じて再帰的に繰り返される。

[0112] 本実施の形態に関わる復号化装置については図8と同一の構成であるため、説明を省略する。

[0113] (ダイレクトモード)

H. 264の多くのフレーム間符号化モードでは、すでに復号化した情報から求められる動き予測に対する動きの予測誤差、および予測で得られたブロックと画像ブロックとの予測残差の2つの情報を符号化および復号化する。しかし、動きの予測誤差を符号化および復号化せず予測残差のみを符号化および復号化するダイレクトモード、すでに復号化した情報から求められる動き予測のみを利用したスキップモードという2つの特殊なモードが用意されている。なお、H. 264ではBスライスと呼ばれるデータに対してのみダイレクトモードが用意されているが、Pスライスと呼ばれるデータについても同じ定義でのダイレクトモードが実現できるため、本実施の形態ではスライスの種類については特に限定しない。

[0114] スキップモードとダイレクトモードという2つのモードを比較すると、スキップモードはダイレクトモードの予測残差が存在しないモードであると考えられる。H. 264では予測残差が存在しないダイレクトモードを符号化

することも可能であるが、そのような場合にはスキップモードで符号化し、復号化時にダイレクトモードでは常に予測残差が存在すると解釈してもよく、予測残差が存在しないダイレクトモードは冗長なモードであることがわかる。したがって、拡張マクロブロックにおける階層的なコードットブロックデータをデコードする際には、ダイレクトモードとして符号化されたブロックと同一サイズのコードットブロックデータに限り、コードットブロックデータをエントロピー復号化により取得せずに、予測残差の存在を表す固定値（１）を設定できる。換言すれば、ブロック単位での予測情報がダイレクトモードである場合に、上位コードットパターンデータを前記ビットストリームから復号化せずに予め定めた値を設定する。なお、ダイレクトモードにおける下位のブロック（例えば、 64×64 画素のレベル２拡張マクロブロックでダイレクトモードと記録されている場合にそのブロックを構成する４つの 32×32 画素の拡張マクロブロック）については、予測残差の存在の有無は不明であって、コードットブロックデータを符号化および復号化する必要がある。

[0115] （スキップフラグ）

今までの説明ではコードットブロックデータの無駄を減らす手段について述べてきた。同じ手段で、スキップモードをあらわすスキップフラグについても冗長な情報を減らすことができる。スキップモードは、ブロックがスキップである場合に１、そうでない場合に０となるフラグである。スキップ予測が可能なブロックのサイズとして２種以上のサイズが用意されている場合には、スキップフラグを階層的に符号化する必要がある。この場合、図１６（単純な階層的スキップフラグの一例）のように実装することになるが、これは今まで述べてきたＣＢＰとＣＢＦの例を論理反転させた表になっており、従来のコードットブロックデータが冗長であった理由と同じ理由で冗長である。したがって、図１７（本実施の形態の階層的スキップフラグの一例）に示すように、下位のスキップフラグとして４つのフラグが符号化および復号化される場合、上位のスキップフラグが０、下位のスキップフラグのうち

第1、第2、第3のスキップフラグが1であるなら、第4の下位のスキップフラグを符号化および復号化せずに、0という固定値を利用できる。

[0116] 階層的なスキップフラグを符号化する方法の流れを表す図を図28に示す。符号化は、次の流れで行われる。

[0117] ステップ1. (S2801) 画像入力部601が画像ブロックを読み込む。

[0118] ステップ2. (S2802) 演算ユニット602がS501で読みこんだブロックに対して符号化すべき予測モードをプログラム用メモリ605から読み込む。予測モードは、スキップフラグ(Skip)の情報を含む。なお、この予測モードは予め決めておく場合もあるし、複数の予測モードのそれぞれでステップS503以降を行い最も良いモードを選択する場合もある。他にも予測モードを設定する変形例はたくさんある。

[0119] ステップ3. (S2803) 演算ユニット602は上位のスキップフラグを符号化して、符号化データ出力部604はビットストリームに出力する。

[0120] ステップ4. (S2804) このステップは次の条件分岐を実行する。演算ユニット602は、上位のスキップフラグが1であれば、ステップ5~9を飛ばす。そうでなければステップ5を実行する。

[0121] ステップ5. (S2805) 演算ユニット602は、スキップフラグ以外の、上位の予測モードを符号化して、符号化データ出力部604からビットストリームに出力する。

[0122] ステップ6. (S2806) ステップ6~8は下位のスキップフラグのうち1ビットを符号化する過程である。このステップでは次の条件分岐を実行する。演算ユニット602は、すでに符号化された下位のスキップフラグが特定のパターンデータである場合(典型的には、最後の1ビットの下位のスキップフラグを符号化する場合であって、かつ、符号化済みの下位のスキップフラグがすべて1である場合)には、次のビットストリームに出力するステップ7を飛ばし、そうでなければステップ7を実行する。

[0123] ステップ7. (S2807) 演算ユニット602は符号化すべき下位のス

キップフラグを符号化して、符号化データ出力部604からビットストリームに出力する。

[0124] ステップ8. (S2808) 演算ユニット602は、スキップフラグ以外の、下位の予測モードを符号化して、符号化データ出力部604からビットストリームに出力する。また、演算ユニット602は、残差情報があるなら、残差情報を符号化して、符号化データ出力部604からビットストリームに出力する。

[0125] ステップ9. (S2809) 演算ユニット602は、(下位のスキップフラグを含めた) 下位のブロックをすべて符号化していればブロックの符号化を終了する。そうでなければ、次の下位のブロックを符号化するために、ステップ6に戻る。S2806からS2809のループは上位のブロックに属する下位のブロックの数に応じて回る回数が決まり、例えば下位のブロックが4つに分割されているなら4回だけ回すことになる。

[0126] エンコーダは、例えば、符号化すべき予測モードを切り替えながらステップ2～9を繰り返し試したうえで、符号化効率が良い予測モードを選択することで、圧縮率を向上させることができる。ただし、デコード可能なビットストリームを出力するために、このような繰り返しは必須ではなく、別の基準で予測モードを決めてもよい。

[0127] 階層的なスキップフラグの復号化の流れは図7とほとんど同じである。図7に対応する、ブロック単位のスキップフラグの復号化の流れの一例を図18に示す。図18の流れは、S702～S708におけるCBPが、S1802～S1808におけるスキップフラグ(図18ではskipと表記)に置き換わる点が図7と異なる。対応するシンタックスは、例えば、図19のようになる。図19は、階層的なスキップフラグに対する本実施の形態を利用したフラグ削減の一例である。

[0128] (四分木分割による再帰構造をもった階層ブロックにおけるスキップフラグ符号化)

図20Aから図20Cまでで説明したような四分木分割による再帰構造に

においても、前述の通り、スキップフラグ或いはダイレクトモードの冗長度を削減可能である。図25にこの場合のシンタクス構造の一例を示す。図25に示される`coding_unit()`は、予測モードなどに関連する情報を符号化する関数である。本関数の基本構造は、図22とほぼ同じである。上位ブロックのスキップフラグを`upper_skip_flag`、4つの下位ブロックのスキップフラグを`lower_skip_flag`と表現している。ここで、図22の説明時と同様、上位スキップフラグは、当該ブロックを分割するかどうかを示す情報と等価である。`currLowerAvailable`は、当該ブロックを分割することが可能かどうかを示す内部変数である。例えば、予め定めた最小分割ブロックサイズよりも当該ブロックを分割したときのブロックサイズが小さくなる場合、下位ブロックがスキップモードを取ることができないため`currLowerAvailable`は0となる。一方、当該ブロックを更に分割することが可能である場合`currLowerAvailable`は1となる。

[0129] `upper_skip_flag`は、上位ブロックがスキップモードかどうかを示すフラグであり、言い換えれば上位ブロックを分割するかどうかを表している。分割する場合は1、分割しない場合は0となる。

[0130] `AvailableSkip`は、当該ブロックがスキップモードを選択できるかどうかを示す1ビットの内部変数である。当該スライスがイントラ予測のみで構成される`I-slice`である場合に、スキップモードのフラグを符号化することは無駄であり、符号量増に繋がるため、上記場合`AvailableSkip`は0に設定される。それ以外の場合には1に設定される。つまり、`DecodedSkipFlag`が0且つ`AvailableSkip`が1の場合のみ`lower_skip_flag`が符号化される。

[0131] `DecodedSkipFlag`は、当該ブロックにおいて`lower_skip_flag`を符号化又は復号するかどうかを示す1ビットの内部変数である。上位スキップフラグが存在した場合（上位ブロックの`upper_skip_flag`が1の場合）で且つ、4つの下位スキップフラグの内の

最初の3つの `lower__skip__flag` が全て0である場合、`DecodedSkipFlag` が1となる。そうでない場合は、`DecodedSkipFlag` は0に設定される。逆にいえば、`DecodedSkipFlag` が1の場合、下位ブロックの最初から3番目までのスキップフラグが0であるため、4番目に相当する下位ブロックのスキップフラグ (`lower__skip__flag`) は自動的に1に設定される。

[0132] (シンタクス分割例)

このようなシンタクス構造をもつことで、四分木分割されたシンタクス構造における下位スキップフラグの冗長性を削減することが可能となる。なお、本実施の形態に関わるシンタクス要素の `upper__skip__flag` と `lower__skip__flag` を別々の関数として分けることも可能である。この例が図26A及び図26Bに記述されている。

[0133] 図26Aでは、`upper__skip__flag` のみを再帰的に符号化するシンタクス構造である。`upper__skip__flag` が0となる場合に、`prediction__unit()` 関数が呼ばれる。一方、図26Bに `prediction__unit()` 関数が示されている。`prediction__unit()` 関数は、`lower__skip__flag` のみを符号化する関数である。以上のように上位ブロックと下位ブロックを分割しても、分割しない図25と同一の機能を実現できる。

[0134] (コーデットブロックフラグとスキップフラグの組み合わせ例)

実施の形態で説明した、図22に示されるコーデットブロックフラグのシンタクス構造と図26に示されるスキップフラグのシンタクス構造を組み合わせることも可能である。図27にこれら2つを組み合わせた場合のシンタクス構造を示す。`coding__unit()` 関数内部で、`prediction__unit()` 関数と `transform__unit()` 関数が呼ばれる構成とすることで、コーデットブロックフラグとスキップフラグの冗長性をそれぞれ削減できる。尚、図中に示される `DecodeSkipTransFlag` は、図22で説明した `DecodeSkipFlag` と同じ機

能を有するが、スキップフラグの冗長度削減時のフラグの名称が被るため、異なるインデックスを与えている。

[0135] (発明の効果)

以上に述べたように、本実施の形態を用いると、階層的なCBPやCBF、およびスキップフラグについて、その冗長性の排除により、圧縮率の高い符号化手段、およびその符号化手段により圧縮されたデータの復号化手段を提供できる。なお、本実施の形態での手段および方法は従来の冗長性を排除するものであって、新しい符号化モードやフラグの追加は行っていない。したがって、従来の符号化方法に対して本実施の形態に記載の方法でフラグの符号化を行うという変更のみを加えた符号化手段を用いるなら、符号量が減ることはあっても増えることはないと期待できる。

[0136] 上述の実施形態の中で示した処理手順に示された指示は、ソフトウェアであるプログラムに基づいて実行されることが可能である。汎用の計算機システムが、このプログラムを予め記憶しておき、このプログラムを読み込むことにより、上述した実施形態の画像復号化方法および画像符号化方法による効果と同様な効果を得ることも可能である。上述の実施形態で記述された指示は、コンピュータに実行させることのできるプログラムとして、磁気ディスク（フレキシブルディスク、ハードディスクなど）、光ディスク（CD-ROM、CD-R、CD-RW、DVD-ROM、DVD±R、DVD±RWなど）、半導体メモリ、又はこれに類する記録媒体に記録される。コンピュータまたは組み込みシステムが読み取り可能な記憶媒体であれば、その記憶形式は何れの形態であってもよい。コンピュータは、この記録媒体からプログラムを読み込み、このプログラムに基づいてプログラムに記述されている指示をCPUで実行させれば、上述した実施形態の画像復号化方法および画像符号化方法と同様な動作を実現することができる。もちろん、コンピュータがプログラムを取得する場合又は読み込む場合はネットワークを通じて取得又は読み込んでもよい。

また、記憶媒体からコンピュータや組み込みシステムにインストールされ

たプログラムの指示に基づきコンピュータ上で稼働しているOS（オペレーティングシステム）や、データベース管理ソフト、ネットワーク等のMW（ミドルウェア）等が本実施形態を実現するための各処理の一部を実行してもよい。

さらに、本願発明における記憶媒体は、コンピュータあるいは組み込みシステムと独立した媒体に限らず、LANやインターネット等により伝達されたプログラムをダウンロードして記憶または一時記憶した記憶媒体も含まれる。

また、記憶媒体は1つに限られず、複数の媒体から本実施形態における処理が実行される場合も、本発明における記憶媒体に含まれ、媒体の構成は何れの構成であってもよい。

[0137] なお、本願発明におけるコンピュータまたは組み込みシステムは、記憶媒体に記憶されたプログラムに基づき、本実施形態における各処理を実行するためのものであって、パソコン、マイコン等の1つからなる装置、複数の装置がネットワーク接続されたシステム等の何れの構成であってもよい。

また、本願発明の実施形態におけるコンピュータとは、パソコンに限らず、情報処理機器に含まれる演算処理装置、マイコン等も含み、プログラムによって本発明の実施形態における機能を実現することが可能な機器、装置を総称している。

[0138] なお、本発明は上記実施形態そのままに限定されるものではなく、実施段階ではその要旨を逸脱しない範囲で構成要素を変形して具体化できる。また、上記実施形態に開示されている複数の構成要素の適宜な組み合わせにより、種々の発明を形成できる。例えば、実施形態に示される全構成要素から幾つかの構成要素を削除してもよい。さらに、異なる実施形態にわたる構成要素を適宜組み合わせてもよい。

符号の説明

[0139] 601…画像入力部、602、802…演算ユニット、603、803…データ用メモリ、604…符号化データ出力部、605、805…プログラム

用メモリ、801…符号化データ入力部、804…画像出力部。

請求の範囲

- [請求項1] 入力画像信号を、階層の深さNに応じたブロックサイズに分割されたブロックを復号する動画像復号化方法において、
- ブロック単位での分割情報を示すフラグを含むビットストリームを取得するステップと、
- 前記階層の深さNよりも深さが小さい階層に属する上位ブロックに付与された、前記上位ブロックの四分木の分割情報を示す上位フラグを復号し取得するステップと、
- 前記階層の深さNに属する下位ブロックのうち、第1の下位ブロックに付与された、前記上位ブロックの四分木の分割情報を示す第1の下位フラグを復号し取得するステップと、
- 既に取得している前記上位フラグ及び複数の前記第1の下位フラグが特定の組み合わせである場合に、前記階層の深さNに属する第2の下位ブロックの四分木の分割情報を示す第2の下位フラグとして予め定めた第1の値を設定し、前記特定の組み合わせではない場合に、前記第2の下位フラグを復号し取得するステップと、
- 前記上位フラグと、第1及び第2の下位フラグに従って、前記ブロックの復号画像信号を生成するステップと、
- を具備する動画像復号化方法。
- [請求項2] 前記フラグは、予測残差信号の有無を表すコーデットブロックフラグであることを、特徴とする請求項1記載の動画像復号化方法。
- [請求項3] 前記フラグは、画面間予測で使われるスキップモードかどうかを表すスキップフラグであることを、特徴とする請求項1記載の動画像復号化方法。
- [請求項4] 前記特定の組み合わせは、前記上位フラグが前記第1の値であり、前記上位ブロックに包含される複数の前記下位ブロックのうち、前記第2の下位ブロックを除いた全ての下位ブロックの前記第1のフラグが前記第1の値とは異なる第2の値であることを、特徴とする請求項

1 記載の動画像復号化方法。

[請求項5]

入力画像信号を、階層の深さNに応じたブロックサイズに分割されたブロックに符号化処理を行う動画像符号化方法において、

ブロック単位での分割情報を示すフラグを符号化するステップと、

前記階層の深さNよりも深さが小さい階層に属する上位ブロックに付与された、前記上位ブロックの四分木の分割情報を示す上位フラグを取得するステップと、

前記階層の深さNに属する下位ブロックのうち、第1の下位ブロックに付与された、前記上位ブロックの四分木の分割情報を示す第1の下位フラグを取得するステップと、

既に取得している前記上位フラグ及び複数の前記第1の下位フラグが特定の組み合わせである場合に、前記階層の深さNに属する第2の下位ブロックの四分木の分割情報を示す第2の下位フラグとして予め定めた第1の値を設定し、前記特定の組み合わせではない場合に、前記第2の下位フラグを取得するステップと、

前記上位フラグと、第1及び第2の下位フラグに従って、前記ブロックの局部復号画像信号を生成するステップと、
を具備する動画像符号化方法。

[請求項6]

前記フラグは、予測残差信号の有無を表すコーデットブロックフラグであることを、特徴とする請求項5記載の動画像符号化方法。

[請求項7]

前記フラグは、画面間予測で使われるスキップモードかどうかを表すスキップフラグであることを、特徴とする請求項5記載の動画像符号化方法。

[請求項8]

記特定の組み合わせは、前記上位フラグが前記第1の値であり、前記上位ブロックに包含される複数の前記下位ブロックのうち、前記第2の下位ブロックを除いた全ての下位ブロックの前記第1のフラグが前記第1の値とは異なる第2の値であることを特徴とする請求項5記載の動画像符号化方法。

[請求項9]

ブロック単位での予測情報、ブロック単位での予測残差信号の有無を表すコーデットパターンデータ、およびブロック単位での予測残差信号をエントロピー符号化した画像データを含むビットストリームを入力するステップと、

前記ビットストリームに含まれる情報が上位コーデットパターンデータである場合に、前記ビットストリームを復号化し上位コーデットパターンデータを取得するステップと、

前記ビットストリームに含まれる情報が前記上位コーデットパターンデータに属するコーデットパターンデータである下位コーデットパターンデータであり、かつ、すでに取得した前記上位コーデットパターンデータと該下位コーデットパターンデータの組み合わせが特定の組み合わせである場合に下位コーデットパターンデータとして予め定めたデータを設定し、前記特定の組み合わせでない場合に前記ビットストリームを復号化し次の下位コーデットパターンデータを取得するステップと、

前記コーデットパターンデータによりブロック単位での予測残差信号の有無を判別するステップと、

前記ビットストリームに含まれる情報がブロック単位での予測情報である場合には、前記ビットストリームを復号化しブロック単位での予測情報を得るステップと、

前記ビットストリームに含まれる情報がブロック単位での予測残差信号である場合には、前記ビットストリームを復号化しブロック単位での予測残差信号を得るステップと、

前記ブロック単位での予測情報、および前記ブロック単位での予測残差信号を利用して、画像信号を復号化するステップと、を具備することを特徴とする画像復号化方法。

[請求項10]

ブロック単位での予測情報、およびブロック単位での予測残差信号をエントロピー符号化した画像データを含むビットストリームを入力

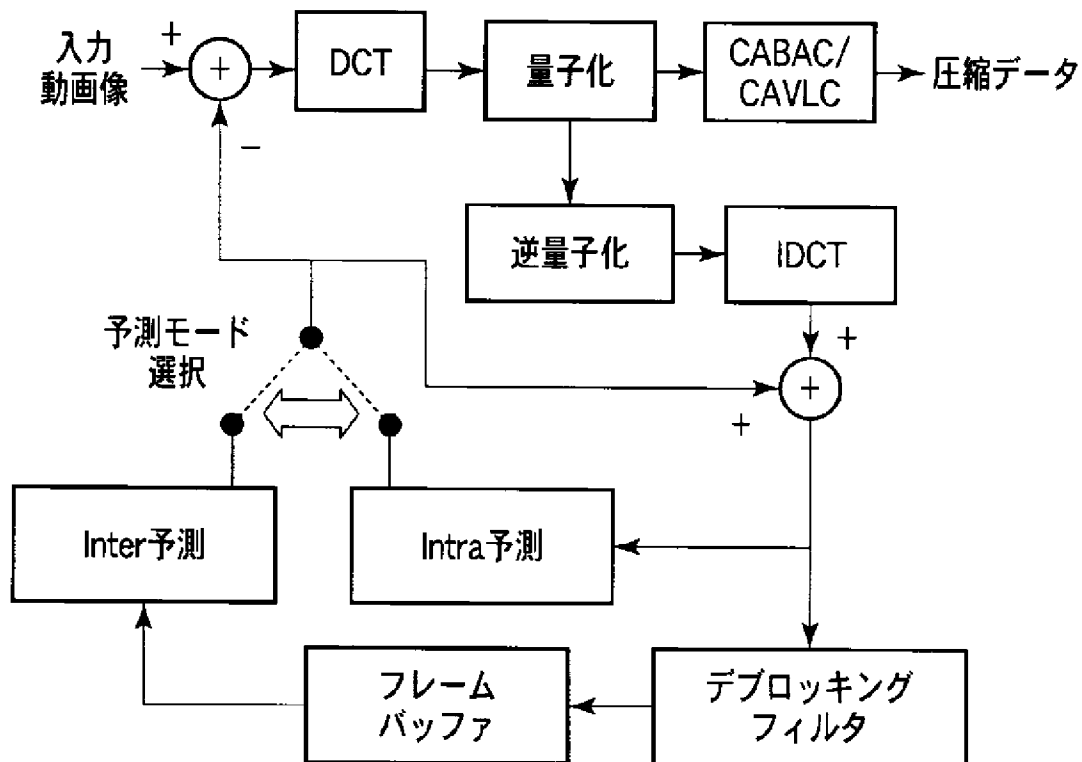
するステップと、

前記ビットストリームに含まれる予測情報が複数のブロックのスキップ予測を表す上位スキップデータである場合に、前記ビットストリームを復号化し上位スキップデータを取得するステップと、

前記ビットストリームに含まれる予測情報が前記上位スキップデータに属するスキップデータである下位スキップデータであり、かつ、すでに取得した前記上位スキップデータと該下位スキップデータの組み合わせが特定の組み合わせである場合に次の下位スキップデータとして予め定めたデータを設定し、前記特定の組み合わせでない場合に前記ビットストリームを復号化し次の下位スキップデータを取得するステップと、

前記ブロック単位での予測情報、および前記ブロック単位での予測残差信号を利用して、画像信号を復号化するステップと、を具備することを特徴とする画像復号化方法。

[図1]



[図2]

lower flags	lower codes	upper flag	lower codesのbit数
0000		0	0
0001	000	1	3
0010	0010	1	4
0011	0011	1	4
:	:	:	:
1110	1110	1	4
1111	1111	1	4

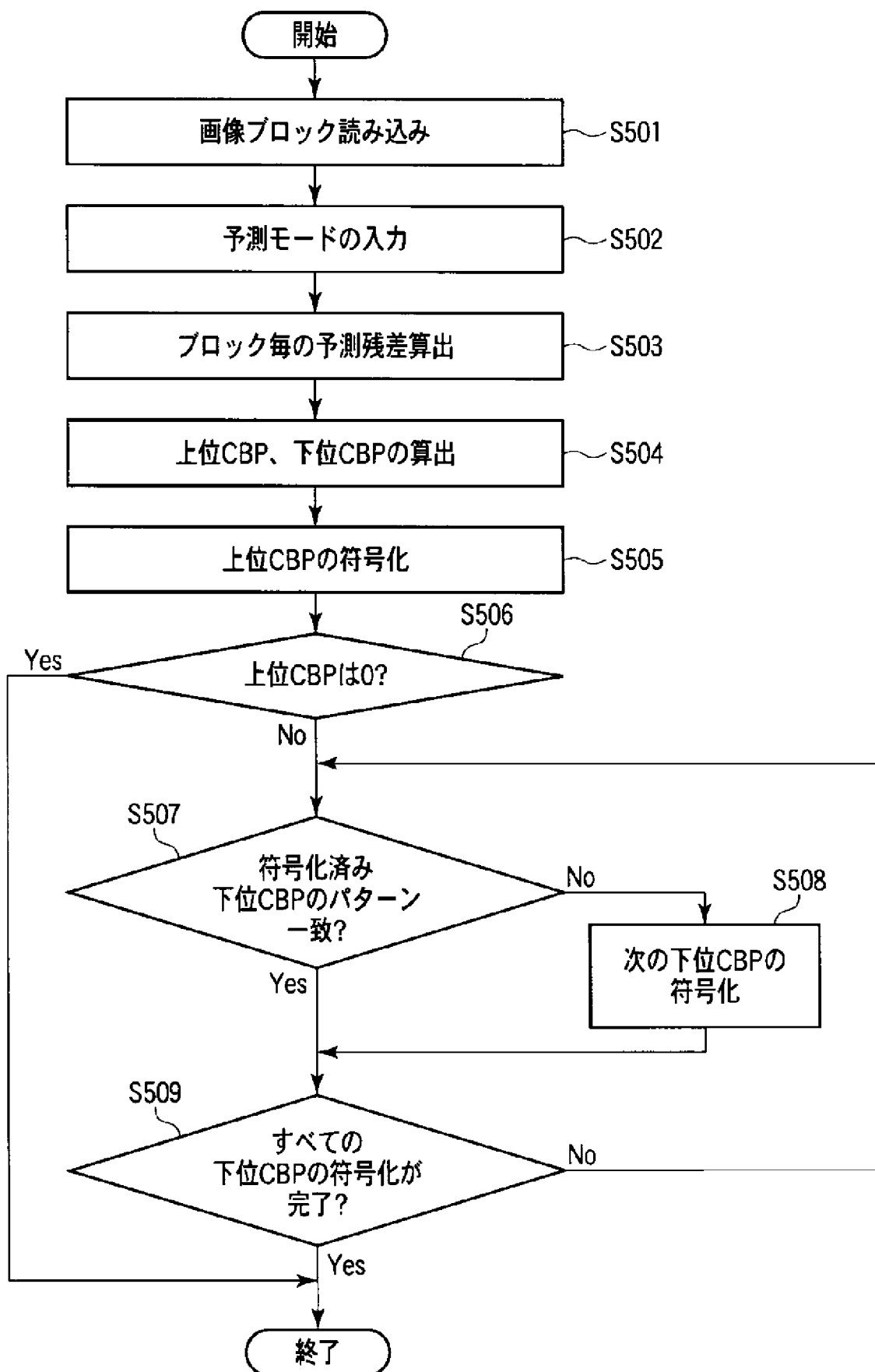
[図3]

lower flags	lower codes	upper flag	lower codesのbit数
0000		0	0
0001	0001	1	4
0010	0010	1	4
0011	0011	1	4
:	:	:	:
1110	1110	1	4
1111	1111	1	4

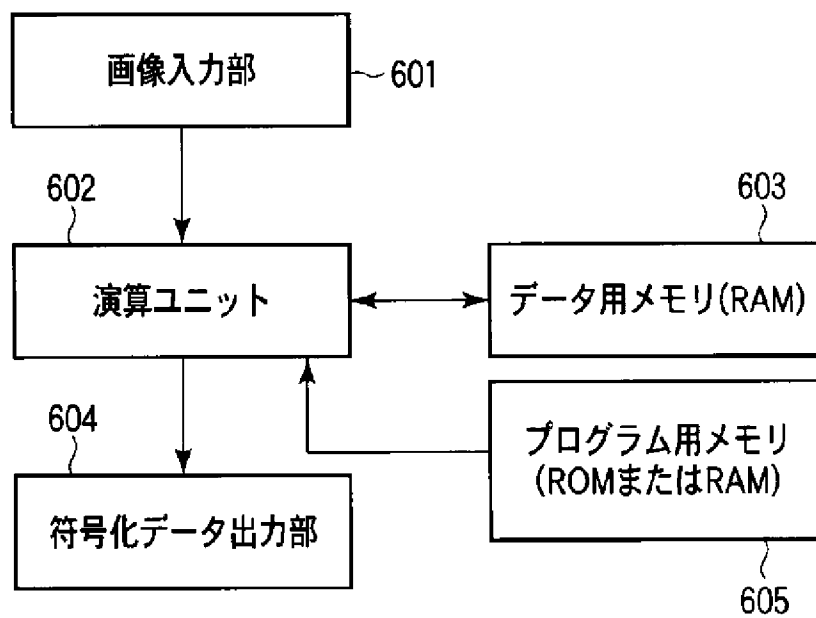
[図4]

lower flags	lower codes	upper flag	lower codesのbit数
00		0	0
01	0	1	1
10	10	1	2
11	11	1	2

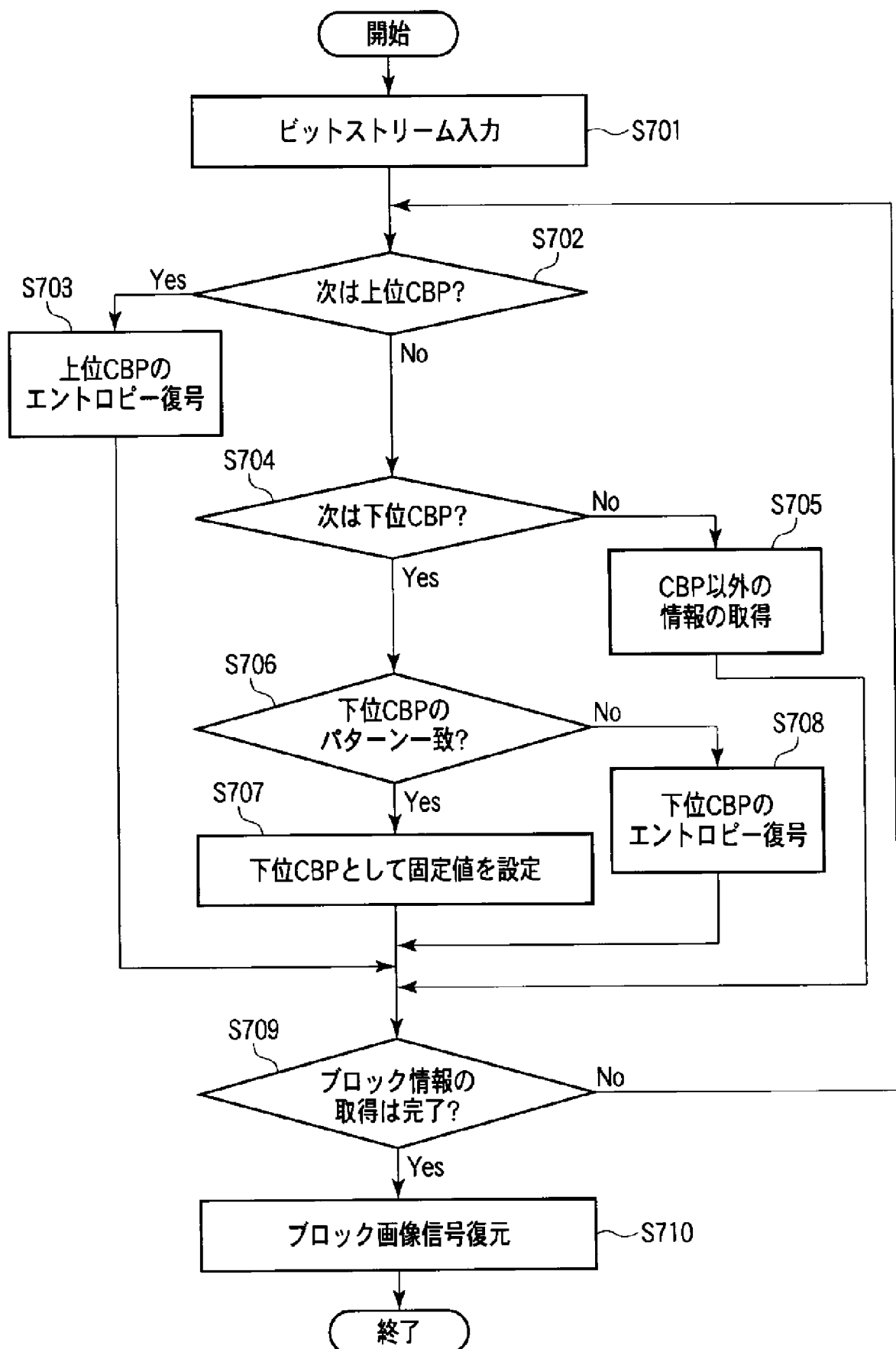
[図5]



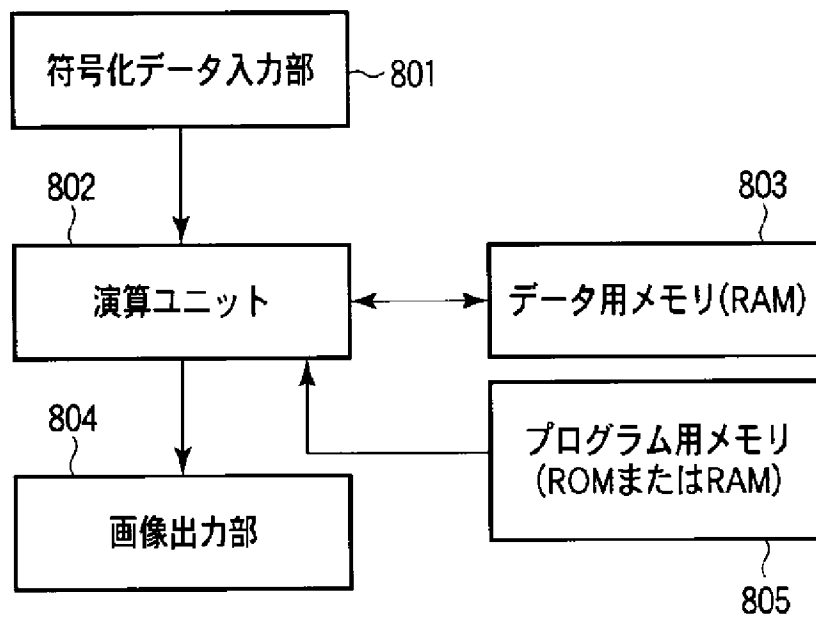
[図6]



[図7]



[図8]



[図9]

(upper_flag is already decoded)	
if (upper_flag != 0) {	
lower_flags[0]	ae(v)
lower_flags[1]	ae(v)
lower_flags[2]	ae(v)
if (lower_flags[0] != 0 lower_flags[1] != 0 lower_flags[2] != 0) {	
lower_flags[3]	ae(v)
} else {	
lower_flags[3] = 1	
}	
(other decoding processes, optional)	
}	

[図10]

(upper_flag is already decoded)	
if (upper_flag != 0) {	
nonzero_lower_flags_found = 0	
for (i = 0; i < num_lower_blocks; i++) {	
if (i != (num_lower_blocks - 1)	
nonzero_lower_flags_found != 0) {	
lower_flags[i]	ae(v)
} else {	
lower_flags[num_lower_blocks - 1] = 1	
}	
if (lower_flags[i] != 0) {	
nonzero_lower_flags_found = 1	
}	
(other decoding processes, optional)	
}	
(other decoding processes, optional)	
}	

[図11]

(upper_flag is already decoded)	
if (upper_flag != 0) {	
lower_flags[0]	ae(v)
lower_flags[1]	ae(v)
lower_flags[2]	ae(v)
if (lower_flags[0] == 0 &&	
lower_flags[1] == 0 &&	
lower_flags[2] == 0) {	
lower_flags[3] = 1	
} else {	
lower_flags[3]	ae(v)
}	
(other decoding processes, optional)	
}	

[図12]

residual_block_with_coded_block_flag_cabac(coeffLevel, startIdx, endIdx, maxNumCoeff, blockIndex) {	
if (blockIndex != 3 nonzero_coded_block_flag_found) {	
coded_block_flag	ae(v)
} else {	
coded_block_flag = 0	
}	
for (i = 0; i < maxNumCoeff; i++)	
coeffLevel[i] = 0	
if (coded_block_flag) {	
nonzero_coded_block_flag_found = 1	
(decode residual coefficients coeffLevel[i])	
}	
}	

[図13]

coded_block_pattern_64	ae(v)
(other decoding processes, optional)	
if (coded_block_pattern_64 != 0) {	
coded_block_pattern_32[0]	ae(v)
(other decoding processes, optional)	
coded_block_pattern_32[1]	ae(v)
(other decoding processes, optional)	
coded_block_pattern_32[2]	ae(v)
(other decoding processes, optional)	
if (coded_block_pattern_32[0] != 0 coded_block_pattern_32[1] != 0 coded_block_pattern_32[2] != 0) {	
coded_block_pattern_32[3]	ae(v)
} else {	
coded_block_pattern_32[3] = 1	
}	
(other decoding processes, optional)	
}	

[図14]

macroblock_cluster_residual_64x64() {	
(other decoding processes, optional)	
coded_block_pattern_64	ae(v)
(other decoding processes, optional)	
if (coded_block_pattern_64 != 0) {	
for (i32x32 = 0; i32x32 < 4; i32x32++)	
macroblock_cluster_residual_32x32(i32x32);	
(other decoding processes, optional)	
}	
(other decoding processes, optional)	
}	
(other decoding processes, optional)	
}	

macroblock_cluster_residual_32x32(block_index) {	
(other decoding processes, optional)	
if (block_index != 3	
coded_block_pattern_32 [0] != 0	
coded_block_pattern_32 [1] != 0	
coded_block_pattern_32 [2] != 0) {	
coded_block_pattern_32 [block_index]	ae(v)
} else {	
coded_block_pattern_32 [3] = 1	
}	
(other decoding processes, optional)	
}	

[図15]

if (cbp_luma_1bit != 0) {	
if (cur_transform_type <= 1) {	
for (cbpldx = 0; cbpldx < 4; cbpldx++) {	
if (cbpldx != 3 cbp != 0) {	
cbp_1bit	ae(v)
} else {	
cbp_1bit = 1	
}	
cbp = cbp_1bit << cbpldx	
}	
} else if (MbPartWidth(mb_type) == 16 && MbPartHeight(mb_type) == 8) {	
for (cbpldx = 0; cbpldx < 2; cbpldx++) {	
if (cbpldx != 1 cbp != 0) {	
cbp_1bit	ae(v)
} else {	
cbp_1bit = 1	
}	
cbp = cbp_1bit << (cbpldx*2)	
}	
cbp = cbp (cbp << 1)	
} else if (MbPartWidth(mb_type) == 8 && MbPartHeight(mb_type) == 16) {	
for (cbpldx = 0; cbpldx < 2; cbpldx++) {	
if (cbpldx != 1 cbp != 0) {	
cbp_1bit	ae(v)
} else {	
cbp_1bit = 1	
}	
cbp = cbp_1bit << cbpldx	
}	
cbp = cbp (cbp << 2)	
} else {	
cbp = cbp_luma_1bit * 0xf / * 16x16 * /	
}	
}	

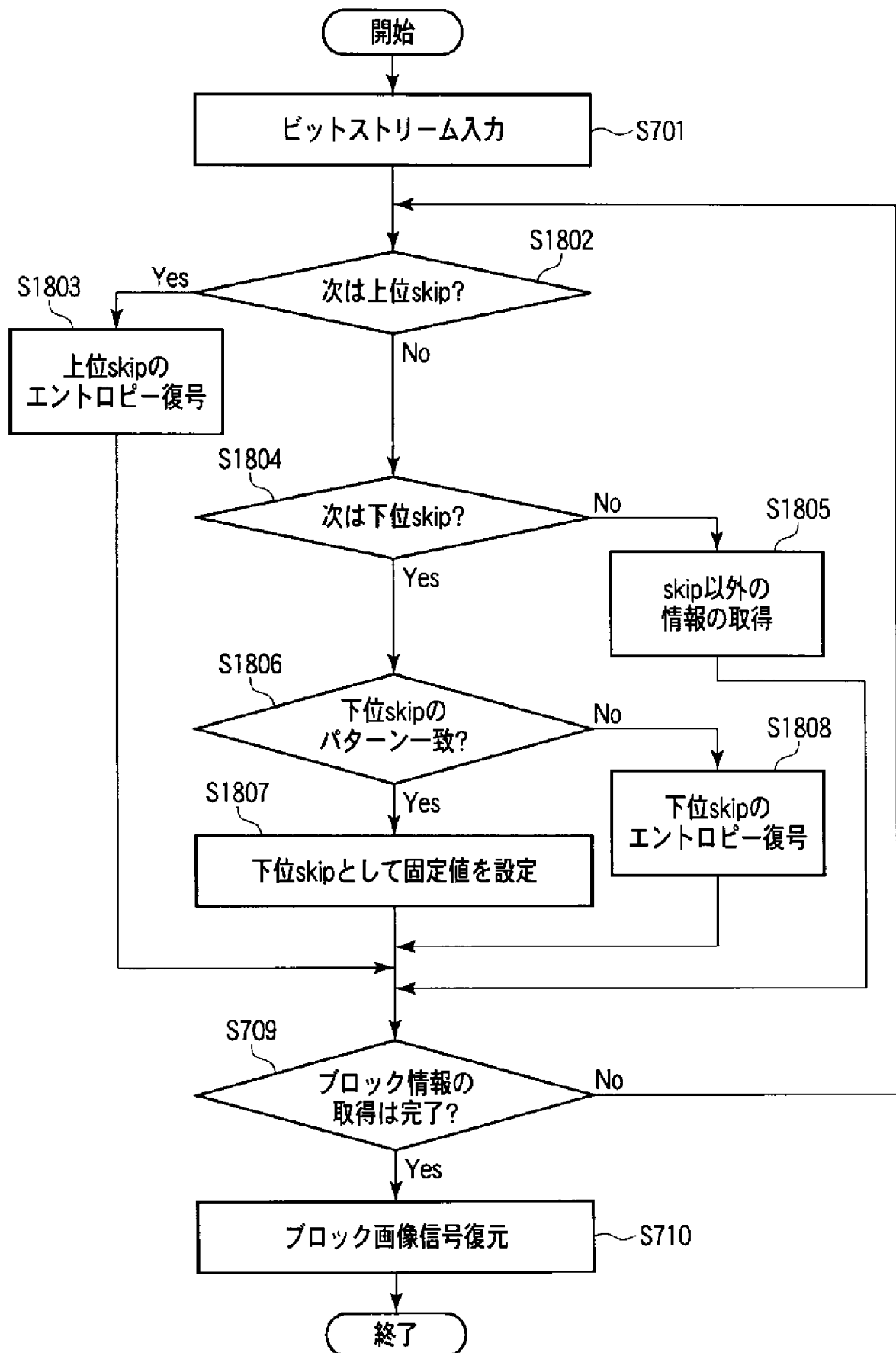
[図16]

lower flags	lower codes	upper flag	lower codesのbit数
1111		1	0
1110	1110	0	4
1101	1101	0	4
1100	1100	0	4
:	:	:	:
0001	0001	0	4
0000	0000	0	4

[図17]

lower flags	lower codes	upper flag	lower codesのbit数
1111		1	0
1110	111	0	3
1101	1101	0	4
1100	1100	0	4
:	:	:	:
0001	0001	0	4
0000	0000	0	4

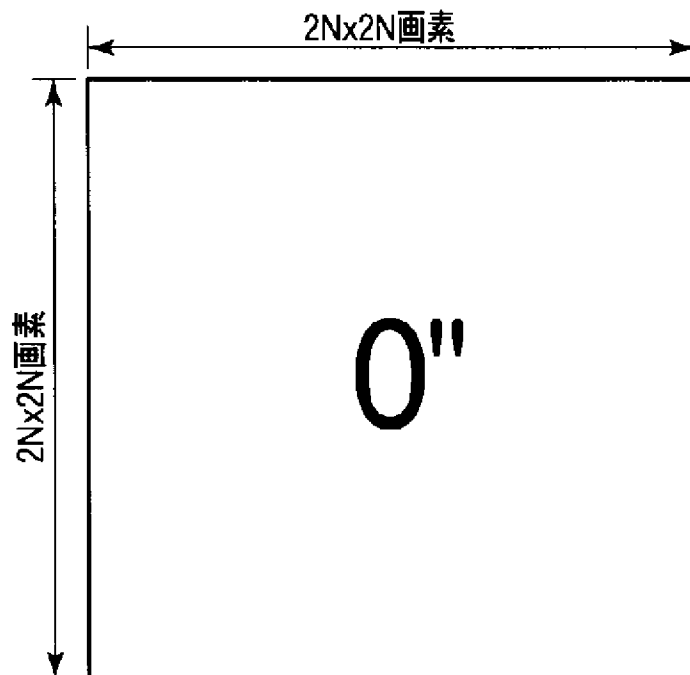
[図18]



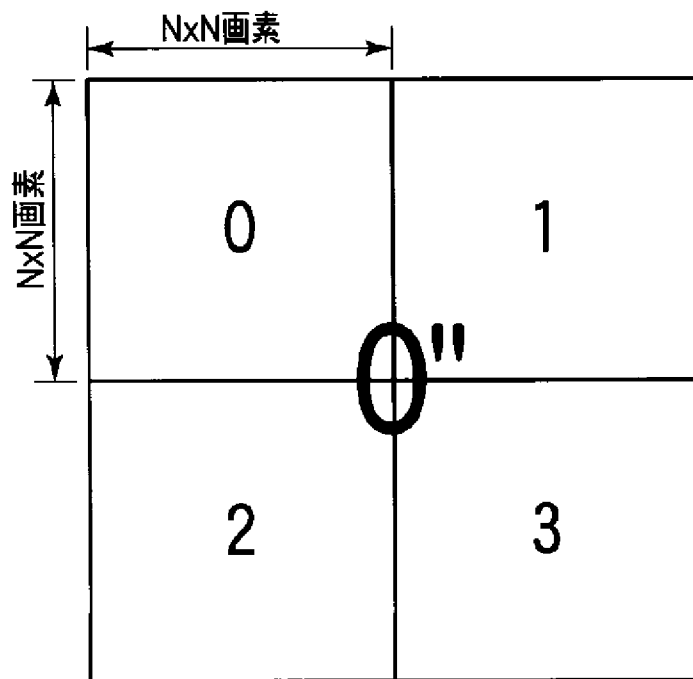
[図19]

mb_skip_flag_64	
if (!mb_skip_flag_64) {	
(decode non-skip stuffs)	
mb_skip_flag_32 [0]	ae(v)
(decode non-skip stuffs)	
mb_skip_flag_32 [1]	ae(v)
(decode non-skip stuffs)	
mb_skip_flag_32 [2]	ae(v)
(decode non-skip stuffs)	
if (mb_skip_flag_32 [0] == 0	
mb_skip_flag_32 [1] == 0	
mb_skip_flag_32 [2] == 0) {	
mb_skip_flag_32 [3]	ae(v)
} else {	
mb_skip_flag_32 [3] = 1	
}	
(decode non-skip stuffs)	
}	

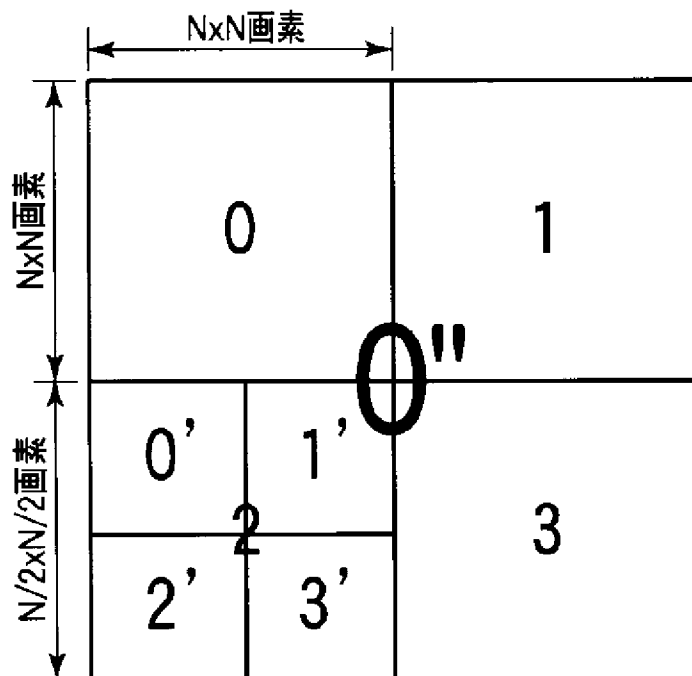
[図20A]



[図20B]



[図20C]



[図21]

transform_unit(currBlockPos, layer) {	Descriptor
if(currLowerAvailable) {	
upper_coded_block_flag	u(1) ae(v)
}	
if(upper_coded_block_flag) {	
currBlockPos0 = calculateBlockPos(currBlockPos, 0)	
transform_unit(currBlockPos0, layer+1)	
currBlockPos1 = calculateBlockPos(currBlockPos, 1)	
transform_unit(currBlockPos1, layer+1)	
currBlockPos2 = calculateBlockPos(currBlockPos, 2)	
transform_unit(currBlockPos2, layer+1)	
currBlockPos3 = calculateBlockPos(currBlockPos, 3)	
transform_unit(currBlockPos3, layer+1)	
} else{	
lower_coded_block_flag	u(1) ae(v)
if(lower_coded_block_flag)	
redisual_block()	
}	
}	

[図22]

transform_unit(currBlockPos, layer, DecodeSkipFlag) {	Descriptor
if(currLowerAvailable) {	
upper_coded_block_flag	u(1) ae(v)
}	
if(upper_coded_block_flag) {	
currBlockPos0 = calculateBlockPos(currBlockPos, 0)	
transform_unit(currBlockPos0, layer+1, DecodeSkipFlag)	
currBlockPos1 = calculateBlockPos(currBlockPos, 1)	
transform_unit(currBlockPos1, layer+1, DecodeSkipFlag)	
currBlockPos2 = calculateBlockPos(currBlockPos, 2)	
transform_unit(currBlockPos2, layer+1, DecodeSkipFlag)	
currBlockPos3 = calculateBlockPos(currBlockPos, 3)	
transform_unit(currBlockPos3, layer+1, DecodeSkipFlag)	
} else{	
if(!DecodeSkipFlag)	
lower_coded_block_flag	u(1) ae(v)
else	
lower_coded_block_flag = 1	
if(lower_coded_block_flag)	
redisual_block()	
}	
}	

[図23A]

transform_unit(currBlockPos, layer, DecodeSkipFlag) {	Descriptor
if(currLowerAvailable) {	
upper_coded_block_flag	u(1) ae(v)
}	
if(upper_coded_block_flag) {	
currBlockPos0 = calculateBlockPos(currBlockPos, 0)	
transform_unit(currBlockPos0, layer+1, DecodeSkipFlag)	
currBlockPos1 = calculateBlockPos(currBlockPos, 1)	
transform_unit(currBlockPos1, layer+1, DecodeSkipFlag)	
currBlockPos2 = calculateBlockPos(currBlockPos, 2)	
transform_unit(currBlockPos1, layer+1, DecodeSkipFlag)	
currBlockPos3 = calculateBlockPos(currBlockPos, 3)	
transform_unit(currBlockPos3, layer+1, DecodeSkipFlag)	
} else{	
(. . .)	
transform_unit_tree(currBlockPos, layer, DecodeSkipFlag)	
(. . .)	
}	
}	

[図23B]

transform_unit_tree(currBlockPos, layer, DecodeSkipFlag) {	Descriptor
if(!DecodeSkipFlag)	
lower_coded_block_flag	u(1) ae(v)
else	
lower_coded_block_flag = 1	
if(lower_coded_block_flag)	
redisual_block()	
}	

[図24A]

BlockSize	layer	Block Number	upper_coded_block_flag	lower_coded_block_flag
2Nx2N	0	0	1	N/A
N/N	1	0	0	0
N/N	1	1	0	0
N/N	1	2	0	0
N/N	1	3	0	1

[図24B]

BlockSize	layer	Block Number	upper_coded_block_flag	lower_coded_block_flag
2Nx2N	0	0	1	N/A
N/N	1	0	0	0
N/N	1	1	0	0
N/N	1	2	0	0
N/N	1	3	0	N/A

[図24C]

BlockSize	layer	Block Number	upper_coded_block_flag	lower_coded_block_flag
2Nx2N	0	0	1	N/A
N/N	1	0	0	0
N/N	1	1	0	0
N/N	1	2	1	N/A
N/2xN/2	2	0	0	0
N/2xN/2	2	1	0	0
N/2xN/2	2	2	0	0
N/2xN/2	2	3	0	1
N/N	1	3	0	0

[図24D]

BlockSize	layer	Block Number	upper_coded_block_flag	lower_coded_block_flag
2Nx2N	0	0	1	N/A
N/N	1	0	0	0
N/N	1	1	0	0
N/N	1	2	1	N/A
N/2xN/2	2	0	0	0
N/2xN/2	2	1	0	0
N/2xN/2	2	2	0	0
N/2xN/2	2	3	0	N/A
NxN	1	3	0	0

[図25]

coding_unit(currBlockPos, layer, DecodeSkipFlag) {	Descriptor
if(currLowerAvailable) {	
upper_skip_flag	u(1) ae(v)
}	
if(upper_skip_flag) {	
currBlockPos0 = calculateBlockPos(currBlockPos, 0)	
coding_unit(currBlockPos0, layer+1, DecodeSkipFlag)	
currBlockPos1 = calculateBlockPos(currBlockPos, 1)	
coding_unit(currBlockPos1, layer+1, DecodeSkipFlag)	
currBlockPos2 = calculateBlockPos(currBlockPos, 2)	
coding_unit(currBlockPos2, layer+1, DecodeSkipFlag)	
currBlockPos3 = calculateBlockPos(currBlockPos, 3)	
coding_unit(currBlockPos3, layer+1, DecodeSkipFlag)	
} else {	
if(AvailableSkip) {	
if(!DecodeSkipFlag)	
lower_skip_flag	u(1) ae(v)
else	
lower_skip_flag = 1	
}	
(. . .)	
}	
}	

[図26A]

coding_unit(currBlockPos, layer, DecodeSkipFlag) {	Descriptor
if(currLowerAvailable) {	
upper_skip_flag	u(1) ae(v)
}	
if(upper_skip_flag) {	
currBlockPos0 = calculateBlockPos(currBlockPos, 0)	
coding_unit(currBlockPos0, layer+1, DecodeSkipFlag)	
currBlockPos1 = calculateBlockPos(currBlockPos, 1)	
coding_unit(currBlockPos1, layer+1, DecodeSkipFlag)	
currBlockPos2 = calculateBlockPos(currBlockPos, 2)	
coding_unit(currBlockPos2, layer+1, DecodeSkipFlag)	
currBlockPos3 = calculateBlockPos(currBlockPos, 3)	
coding_unit(currBlockPos3, layer+1, DecodeSkipFlag)	
} else {	
(. . .)	
prediction_unit(currBlockPos, layer, DecodeSkipFlag)	
(. . .)	
}	
}	

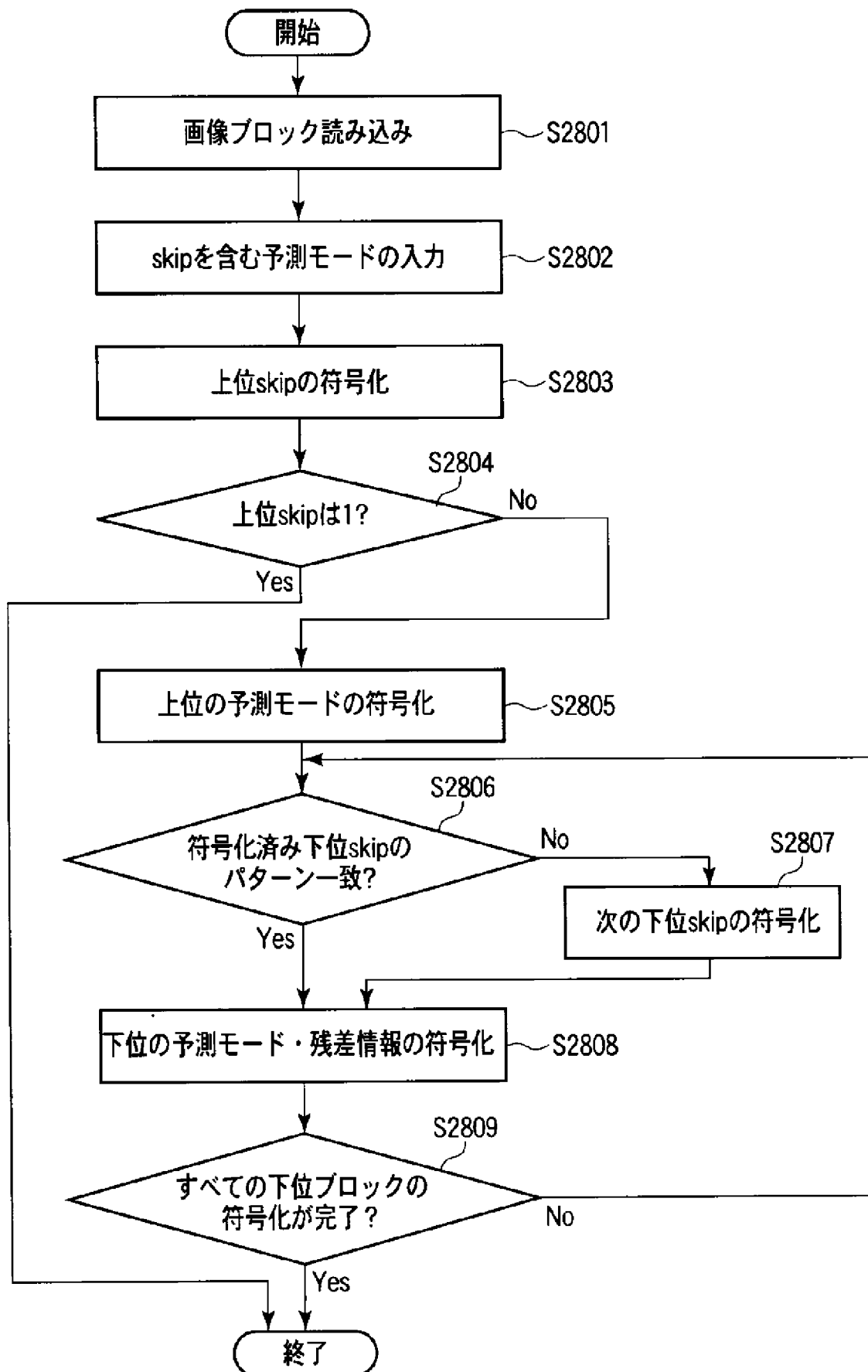
[図26B]

prediction_unit(currBlockPos, layer, DecodeSkipFlag) {	Descriptor
if(AvailableSkip) {	
if(!DecodeSkipFlag)	
lower_skip_flag	u(1) ae(v)
else	
lower_skip_flag = 1	
}	
(. . .)	
}	

[図27]

coding_unit(currBlockPos, layer, DecodeSkipFlag) {	Descriptor
if(currLowerAvailable) {	
upper_skip_flag	u(1) ae(v)
}	
if(upper_skip_flag) {	
currBlockPos0 = calculateBlockPos(currBlockPos, 0)	
coding_unit(currBlockPos0, layer+1, DecodeSkipFlag)	
currBlockPos1 = calculateBlockPos(currBlockPos, 1)	
coding_unit(currBlockPos1, layer+1, DecodeSkipFlag)	
currBlockPos2 = calculateBlockPos(currBlockPos, 2)	
coding_unit(currBlockPos2, layer+1, DecodeSkipFlag)	
currBlockPos3 = calculateBlockPos(currBlockPos, 3)	
coding_unit(currBlockPos3, layer+1, DecodeSkipFlag)	
} ekse {	
(. . .)	
prediction_unit(currBlockPos, layer, DecodeSkipFlag)	
transform_unit(currBlockPos, layer, DecodeSkipTransFlag)	
(. . .)	
}	
}	

[図28]



INTERNATIONAL SEARCH REPORT

International application No.

PCT/JP2010/071155

A. CLASSIFICATION OF SUBJECT MATTER

H04N7/32 (2006.01) i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

H04N7/32

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Jitsuyo Shinan Koho	1922-1996	Jitsuyo Shinan Toroku Koho	1996-2011
Kokai Jitsuyo Shinan Koho	1971-2011	Toroku Jitsuyo Shinan Koho	1994-2011

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JP 2009-531997 A (Qualcomm Inc.), 03 September 2009 (03.09.2009), entire text; all drawings & US 2007/0223579 A1 & EP 1999971 A & WO 2007/112417 A2	1-10
A	JP 2009-510962 A (Nokia Corp.), 12 March 2009 (12.03.2009), paragraph [0017] & WO 2007/010374 A1	1-10
A	JP 2003-324731 A (Sony Corp.), 14 November 2003 (14.11.2003), paragraphs [0037] to [0045] (Family: none)	1-10

☐ Further documents are listed in the continuation of Box C.

☐ See patent family annex.

* Special categories of cited documents:

“A” document defining the general state of the art which is not considered to be of particular relevance

“E” earlier application or patent but published on or after the international filing date

“L” document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

“O” document referring to an oral disclosure, use, exhibition or other means

“P” document published prior to the international filing date but later than the priority date claimed

“T” later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

“X” document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

“Y” document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

“&” document member of the same patent family

Date of the actual completion of the international search
17 January, 2011 (17.01.11)

Date of mailing of the international search report
25 January, 2011 (25.01.11)

Name and mailing address of the ISA/
Japanese Patent Office

Authorized officer

Facsimile No.

Telephone No.

A. 発明の属する分野の分類 (国際特許分類 (I P C))

Int.Cl. H04N7/32(2006.01)i

B. 調査を行った分野

調査を行った最小限資料 (国際特許分類 (I P C))

Int.Cl. H04N7/32

最小限資料以外の資料で調査を行った分野に含まれるもの

日本国実用新案公報	1 9 2 2 - 1 9 9 6 年
日本国公開実用新案公報	1 9 7 1 - 2 0 1 1 年
日本国実用新案登録公報	1 9 9 6 - 2 0 1 1 年
日本国登録実用新案公報	1 9 9 4 - 2 0 1 1 年

国際調査で使用した電子データベース (データベースの名称、調査に使用した用語)

C. 関連すると認められる文献

引用文献の カテゴリー*	引用文献名 及び一部の箇所が関連するときは、その関連する箇所の表示	関連する 請求項の番号
A	JP 2009-531997 A (クウアルコム・インコーポレイテッド) 2009.09.03, 全文全図 & US 2007/0223579 A1 & EP 1999971 A & WO 2007/112417 A2	1-10
A	JP 2009-510962 A (ノキアコーポレイション) 2009.03.12, 【00 17】 & WO 2007/010374 A1	1-10
A	JP 2003-324731 A (ソニー株式会社) 2003.11.14, 【0037】 - 【0045】 (ファミリーなし)	1-10

C欄の続きにも文献が列挙されている。

パテントファミリーに関する別紙を参照。

* 引用文献のカテゴリー

「A」特に関連のある文献ではなく、一般的技術水準を示すもの
「E」国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの
「L」優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献 (理由を付す)
「O」口頭による開示、使用、展示等に言及する文献
「P」国際出願日前で、かつ優先権の主張の基礎となる出願

の日の後に公表された文献

「T」国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの
「X」特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの
「Y」特に関連のある文献であって、当該文献と他の1以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの
「&」同一パテントファミリー文献

国際調査を完了した日

1 7 . 0 1 . 2 0 1 1

国際調査報告の発送日

2 5 . 0 1 . 2 0 1 1

国際調査機関の名称及びあて先

日本国特許庁 (I S A / J P)

郵便番号 1 0 0 - 8 9 1 5

東京都千代田区霞が関三丁目4番3号

特許庁審査官 (権限のある職員)

古市 徹

電話番号 0 3 - 3 5 8 1 - 1 1 0 1 内線 3 5 4 1

5 C

3 0 5 3