(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3870171号
(P3870171)
(45)発行日　平成19年1月17日(2007.1.17)　(24)登録日　平成18年10月20日(2006.10.20)

(51)Int.Cl.			F I		
H03M	7/46	(2006.01)	H03M	7/46	
H04N	1/419	(2006.01)	H04N	1/419	
H04N	7/26	(2006.01)	H04N	7/13	Z

請求項の数 8　(全 35 頁)

(21)出願番号　特願2003-64778(P2003-64778)
(22)出願日　平成15年3月11日(2003.3.11)
(65)公開番号　特開2004-274554(P2004-274554A)
(43)公開日　平成16年9月30日(2004.9.30)
審査請求日　平成17年1月11日(2005.1.11)

(73)特許権者　000001007
キヤノン株式会社
東京都大田区下丸子３丁目３０番２号
(74)代理人　100076428
弁理士　大塚　康徳
(74)代理人　100112508
弁理士　高柳　司郎
(74)代理人　100115071
弁理士　大塚　康弘
(74)代理人　100116894
弁理士　木村　秀二
(72)発明者　梅田　嘉伸
東京都大田区下丸子３丁目３０番２号　キヤノン株式会社内

最終頁に続く

(54)【発明の名称】符号化方法及び符号化装置、コンピュータプログラム並びにコンピュータ可読記憶媒体

(57)【特許請求の範囲】

【請求項１】

同じデータの連続数を示す連続数コード部と前記データを示すデータ部とで表される第１のタイプの符号化データ、又は、異なるデータの連続数を示す連続コード部と前記異なるデータのデータ部で表わされる第２のタイプの符号化データを順次入力し、前記第１又は第２のタイプの符号化データ形式の符号化データに再符号化する符号化方法であって、

入力した符号化データの連続数コード部に基づき、注目符号化データが前記第１、第２のタイプのいずれの符号化データであるかを判定する判定工程と、

該判定工程で注目符号化データが前記第１のタイプの符号化データであると判定した場合には、注目符号化データ中の連続数コード部で示される連続個数情報と、データ部で示されるデータを出力し、

該判定工程で注目符号化データが前記第２のタイプの符号化データであると判定した場合には、注目符号化データ中のデータ部の個々のデータを出力する際、同じデータの連続個数が“１”であることを示す連続個数情報を出力する符号化データ分離工程と、

前記符号化データ分離工程より出力された連続個数情報及びデータのうち、前記データの一部のビットを所定値に変更し、出力するデータ変更工程と、

前記符号化データ分離工程より出力された連続個数情報、及び、前記データ変更工程で処理されたデータとを入力し、復号処理することなく、前記第１又は第２のタイプの符号化データの連続数コード部とデータ部を再構成する再構成工程と、

再構成された連続数コード部とデータ部を再符号化データとして出力する工程とを備え

、
前記再構成工程は、
前記符号化データ分離工程からの連続個数情報と前記データ変更工程からの変更後のデータの組を入力する毎に、同じデータの連続数を計数中か異なるデータの連続数を計数中か、及び、現入力のデータが直前に入力されたデータとが等しいか否かを判断する判断工程と、
該判断工程で、同じデータの連続数を計数中であり、且つ、現入力のデータが直前に入力されたデータと等しい場合、現入力の連続個数情報に、直前の入力で更新された連続個数情報を加算することで、現連続個数情報を更新し、
前記判断工程で、異なるデータの連続数を計数中であり、且つ、現入力のデータが直前に入力されたデータと等しくない場合、現入力の連続個数情報に、直前の入力で更新された連続個数情報を加算することで、現連続個数情報を更新し、
前記判断工程で、同じデータの連続数を計数中であり、且つ、現入力のデータが直前に入力されたデータと等しくない場合、又は、異なるデータの連続数を計数中であり、且つ、現入力のデータが直前に入力されたデータと等しい場合、計数中であった未出力のデータからデータ部、計数された連続数個数情報から連続数コード部を生成する工程と
を備えることを特徴とする符号化方法。
【請求項２】
更に、前記データ変更工程に対し、変更するビット位置を指定する工程を有することを特徴とする請求項１に記載の符号化方法。
【請求項３】
前記符号化データのデータ部は、画像データの画素の複数の像域属性のフラグビットで構成されることを特徴とする請求項１又は２に記載の符号化方法。
【請求項４】
前記符号化データのデータ部は、画像データを符号化したデータであることを特徴とする請求項１又は２に記載の符号化方法。
【請求項５】
前記符号化データは、パックビッツ符号化データであることを特徴とする請求項１乃至４のいずれか１項に記載の符号化方法。
【請求項６】
同じデータの連続数を示す連続数コード部と前記データを示すデータ部とで表される第１のタイプの符号化データ、又は、異なるデータの連続数を示す連続コード部と前記異なるデータのデータ部で表わされる第２のタイプの符号化データを順次入力し、前記第１又は第２のタイプの符号化データ形式の符号化データに再符号化する符号化装置であって、
入力した符号化データの連続数コード部に基づき、注目符号化データが前記第１、第２のタイプのいずれの符号化データであるかを判定する判定手段と、
該判定手段で注目符号化データが前記第１のタイプの符号化データであると判定した場合には、注目符号化データ中の連続数コード部で示される連続個数情報と、データ部で示されるデータを出力し、
該判定手段で注目符号化データが前記第２のタイプの符号化データであると判定した場合には、注目符号化データ中のデータ部の個々のデータを出力する際、同じデータの連続個数が“１”であることを示す連続個数情報を出力する符号化データ分離手段と、
前記符号化データ分離手段より出力された連続個数情報及びデータのうち、前記データの一部のビットを所定値に変更し、出力するデータ変更手段と、
前記符号化データ分離手段より出力された連続個数情報、及び、前記データ変更工程で処理されたデータとを入力し、復号処理することなく、前記第１又は第２のタイプの符号化データの連続数コード部とデータ部を再構成する再構成手段と、
再構成された連続数コード部とデータ部を再符号化データとして出力する手段とを備え
、
前記再構成手段は、

前記符号化データ分離手段からの連続個数情報と前記データ変更手段からの変更後のデータの組を入力する毎に、同じデータの連続数を計数中か異なるデータの連続数を計数中か、及び、現入力のデータが直前に入力されたデータとが等しいか否かを判断する判断手段と、

該判断手段で、同じデータの連続数を計数中であり、且つ、現入力のデータが直前に入力されたデータと等しい場合、現入力の連続個数情報に、直前の入力で更新された連続個数情報を加算することで、現連続個数情報を更新し、

前記判断手段で、異なるデータの連続数を計数中であり、且つ、現入力のデータが直前に入力されたデータと等しくない場合、現入力の連続個数情報に、直前の入力で更新された連続個数情報を加算することで、現連続個数情報を更新し、

前記判断手段で、同じデータの連続数を計数中であり、且つ、現入力のデータが直前に入力されたデータと等しくない場合、又は、異なるデータの連続数を計数中であり、且つ、現入力のデータが直前に入力されたデータと等しい場合、計数中であった未出力のデータからデータ部、計数された連続数個数情報から連続数コード部を生成する手段と

を備えることを特徴とする符号化装置。

【請求項７】

コンピュータが読込み実行することで、同じデータの連続数を示す連続数コード部と前記データを示すデータ部とで表される第１のタイプの符号化データ、又は、異なるデータの連続数を示す連続コード部と前記異なるデータのデータ部で表わされる第２のタイプの符号化データを順次入力し、前記第１又は第２のタイプの符号化データ形式の符号化データに再符号化する符号化装置として機能するコンピュータプログラムであって、

入力した符号化データの連続数コード部に基づき、注目符号化データが前記第１、第２のタイプのいずれの符号化データであるかを判定する判定手段と、

該判定手段で注目符号化データが前記第１のタイプの符号化データであると判定した場合には、注目符号化データ中の連続数コード部で示される連続個数情報と、データ部で示されるデータを出力し、

該判定手段で注目符号化データが前記第２のタイプの符号化データであると判定した場合には、注目符号化データ中のデータ部の個々のデータを出力する際、同じデータの連続個数が“１”であることを示す連続個数情報を出力する符号化データ分離手段と、

前記符号化データ分離手段より出力された連続個数情報及びデータのうち、前記データの一部のビットを所定値に変更し、出力するデータ変更手段と、

前記符号化データ分離手段より出力された連続個数情報、及び、前記データ変更工程で処理されたデータとを入力し、復号処理することなく、前記第１又は第２のタイプの符号化データの連続数コード部とデータ部を再構成する再構成手段と、

再構成された連続数コード部とデータ部を再符号化データとして出力する手段とを備え、

前記再構成手段は、

前記符号化データ分離手段からの連続個数情報と前記データ変更手段からの変更後のデータの組を入力する毎に、同じデータの連続数を計数中か異なるデータの連続数を計数中か、及び、現入力のデータが直前に入力されたデータとが等しいか否かを判断する判断手段と、

該判断手段で、同じデータの連続数を計数中であり、且つ、現入力のデータが直前に入力されたデータと等しい場合、現入力の連続個数情報に、直前の入力で更新された連続個数情報を加算することで、現連続個数情報を更新し、

前記判断手段で、異なるデータの連続数を計数中であり、且つ、現入力のデータが直前に入力されたデータと等しくない場合、現入力の連続個数情報に、直前の入力で更新された連続個数情報を加算することで、現連続個数情報を更新し、

前記判断手段で、同じデータの連続数を計数中であり、且つ、現入力のデータが直前に入力されたデータと等しくない場合、又は、異なるデータの連続数を計数中であり、且つ、現入力のデータが直前に入力されたデータと等しい場合、計数中であった未出力のデ

ータからデータ部、計数された連続数個数情報から連続数コード部を生成する手段
としてコンピュータに機能させることを特徴とするコンピュータプログラム。
【請求項８】
請求項７に記載のコンピュータプログラムを格納したことを特徴とするコンピュータ可読記憶媒体。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は符号化データを処理し、再符号化する技術に関するものである。
【０００２】
【従来の技術】
データの圧縮符号化技術としてパックビッツ符号化が知られている。このパックビッツ符号化とはランレングス符号化の１つであり、符号化前の情報を損失することなし復号可能、すなわち、ロスレス符号化でもある。
【０００３】
以下、パックビッツ符号化の説明を、符号化を行う単位であるひとつのデータが８ビットで構成されているものとして行う（データとしては例えば８ビット多値画素データである）。また、符号化する以前のデータを生データと呼び、符号化以前の生データのストリームを生データ列と呼ぶ。
【０００４】
パックビッツ符号化とは、生データ列を、「同じデータが連続する部分」と「異なるデータが連続する部分」とに区別し、それぞれの部分に対してデータの「長さ情報」を付加するものである。そして、「同じデータが連続する部分」に関しては、「連続する数＋データ」という形に、また「異なるデータが連続する部分」に関しては「異なるデータが連続する数＋異なるデータ列」という形で表す。そしてこの処理で生成したデータ列をパックビッツ符号化データとして出力する。
【０００５】
今、以下の（１）ような生データ列（２４個のデータ）が入力されたとする。個々のデータは８ビット（０～２５５の範囲のデータ）。
0,0,0,0,1,2,3,4,4,4,4,4,4,4,4,5,5,6,7,8,8,9,10,10　　…　（１）
この場合、生データ列を
0,0,0,0,
1,2,3,
4,4,4,4,4,4,4,4,
5,5,
6,7,
8,8,
9,
10,10　　…　（２）
というように、「同じデータが連続する部分」と、「異なるデータが連続する部分」とに区別する。そして、それぞれの部分に「長さ情報」を付加し、「同じデータが連続する部分」に関してはデータ部分を1つだけに圧縮する。
【０００６】
0,0,0,0,　　⇒　(-4),0,
1,2,3,　　⇒　(3),1,2,3,
4,4,4,4,4,4,4,4,　⇒　(-8),4,
5,5,　　⇒　(-2),5,
6,7,　　⇒　(2),6,7,
8,8,　　⇒　(-2),8,
9,　　⇒　(1),9,

10,10 ⇒ (-2),10 … （３）

（３）では「長さ情報」は括弧で囲んで示している。「長さ情報」を正負の値で示したのは、連続する部分が同じデータであるのか、異なるデータであるのかを区別するためである。すなわち、負の数は「同じデータの連続部分」を表し、正の数は「異なるデータの連続部分」を表している。そして、この「長さ情報」をデータと同じく８ビットの幅を持つ「長さコードデータ」としてデータ列に埋め込んで出力する。すなわち、長さコードデータの８ビットの最上位ビットＭＳＢを正負の符号ビットとして扱う。

【０００７】

また、上記の値「９」のように１つだけ孤立して存在するデータについては、「異なるデータが１つある」と定義する。そのため、同じデータの連続する数は最低でも２つ以上となる。

【０００８】

長さコードデータと、「同じ値のデータが連続する数」、及び、「異なる値のデータが連続する数」との関係（テーブル）は次のようになる（注：「０ｘ」は１６進数を示す）。

【０００９】

異なるデータが連続する数	長さコードデータ
１	０ｘ００
２	０ｘ０１
３	０ｘ０２
：	：
１２７	０ｘ７Ｅ
１２８	０ｘ７Ｆ

同じデータが連続する数	長さコードデータ
２（－２）	０ｘＦＦ
３（－３）	０ｘＦＥ
：	：
１２７（－１２７）	０ｘ８２
１２８（－１２８）	０ｘ８１

したがって、先に示した生データ列（１）をパックビッツ符号化すると、次の示す符号化データ列（４）が生成されることになる。

FD,00,02,01,02,03,F9,04,FF,5,01,06,07,FF,08,00,09,FF,10 … （４）

上記の如く、パックビッツ符号化では、同じデータの連続する数は２乃至１２８まで表現でき、異なるデータの連続する数は１乃至１２８まで表現できる。従って、仮に、同じデータが連続する数が１２９以上の場合には、１２８個まで連続する符号（長さコードデータ＋連続するデータ）を生成し、１２９番目以降については、その１２９番目のデータを起点とする符号を生成することになる。これは異なるデータの連続する場合にも同様である。

【００１０】

尚、上記表に示すようにパックビッツ符号化では「長さコードデータ」として０ｘ８０を生成しない。従って、この「０ｘ８０」を、符号化データ列の最後を表すエンドオブリストコードデータとして定義することもある。

【００１１】

以降、パックビッツ符号化データにおいて「長さコードデータ」のことを「長さ情報」、長さ情報に続いて出力される実際のデータの値を示す部分を「データ部」という言葉で表現することとする。

【００１２】

以上のように、パックビッツ符号化では同じ値のデータが連続して現れる回数が増加すると符号化後のデータ量が減少する（圧縮率が高くなる）。

【００１３】

【発明が解決しようとする課題】

一般に、データを圧縮する場合、生データの量（サイズ）に対して、１／２以下（圧縮率２倍以上）とか、１／４以下（圧縮率４倍以上）といった目標を立てて圧縮符号化を行うことが多い。パックビッツ符号化の場合、同じデータが３つ以上連続する場合に圧縮率が向上することになるが、異なるデータが連続することが多い場合には「長さ情報」が付加される分だけデータ量が増えることになる。

【００１４】

従って、パックビッツ符号化を行って目標とするデータ量に到達できない場合には、同じデータが連続するように変更を加えない限り、それを実現できない。　ここで、生データの或るビットを０に固定すると、取り得る値の種類数は半分と少なくなり、必然的に同じ値が連続する確率が高くなるので、このようなデータを圧縮した場合は高い圧縮率となる。例えば、圧縮符号化対象として多値画素を例にすると、ＬＳＢを０にした場合、２ｎ、２ｎ＋１（ｎは０、１、２…）で表現される画素値は共に「２ｎ」に丸められる。中間調画像の場合にはもともと隣接する画素の濃度や輝度はその差が小さいので、上記のようにビット値を変更することで更に同じ値になり易くなる。即ち、上記ビット変更処理を行えば、より高い圧縮率になることがわかるであろう。同様に、適当な２ビットを０にすると、更に高い圧縮率となるのは理解できよう。

【００１５】

しかしながら、オリジナルとなる生データの集合（例えば多値画像データ）の１つ１つのデータに上記の丸め処理を行うには、そのデータ量に比例した処理が必要になり、高い処理速度は望めない。また、そのためにオリジナルの生データ全体を記憶しておくメモリも必要とする。

【００１６】

そこで、本願発明は、パックビッツ符号化データの如く、同じデータの連続数を示す連続数コード部と前記データを示すデータ部、及び、異なるデータ列の連続数を示す連続数コード部と前記異なるデータ列を示すデータ部のデータ形式で表現される符号化データを、復号することなく、同じデータ形式に、より高い圧縮率で再符号化する技術を提供しようとするものである。

【００１７】

【課題を解決するための手段】

かかる課題を解決するため、例えば本発明の符号化方法は以下の工程を備える。すなわち、

同じデータの連続数を示す連続数コード部と前記データを示すデータ部とで表される第１のタイプの符号化データ、又は、異なるデータの連続数を示す連続コード部と前記異なるデータのデータ部で表わされる第２のタイプの符号化データを順次入力し、前記第１又は第２のタイプの符号化データ形式の符号化データに再符号化する符号化方法であって、

入力した符号化データの連続数コード部に基づき、注目符号化データが前記第１、第２のタイプのいずれの符号化データであるかを判定する判定工程と、

該判定工程で注目符号化データが前記第１のタイプの符号化データであると判定した場合には、注目符号化データ中の連続数コード部で示される連続個数情報と、データ部で示されるデータを出力し、

該判定工程で注目符号化データが前記第２のタイプの符号化データであると判定した場合には、注目符号化データ中のデータ部の個々のデータを出力する際、同じデータの連続個数が“１”であることを示す連続個数情報を出力する符号化データ分離工程と、

前記符号化データ分離工程より出力された連続個数情報及びデータのうち、前記データの一部のビットを所定値に変更し、出力するデータ変更工程と、

前記符号化データ分離工程より出力された連続個数情報、及び、前記データ変更工程で処理されたデータとを入力し、復号処理することなく、前記第１又は第２のタイプの符号化データの連続数コード部とデータ部を再構成する再構成工程と、

再構成された連続数コード部とデータ部を再符号化データとして出力する工程とを備え、

前記再構成工程は、

前記符号化データ分離工程からの連続個数情報と前記データ変更工程からの変更後のデータの組を入力する毎に、同じデータの連続数を計数中か異なるデータの連続数を計数中か、及び、現入力のデータが直前に入力されたデータとが等しいか否かを判断する判断工程と、

該判断工程で、同じデータの連続数を計数中であり、且つ、現入力のデータが直前に入力されたデータと等しい場合、現入力の連続個数情報に、直前の入力で更新された連続個数情報を加算することで、現連続個数情報を更新し、

前記判断工程で、異なるデータの連続数を計数中であり、且つ、現入力のデータが直前に入力されたデータと等しくない場合、現入力の連続個数情報に、直前の入力で更新された連続個数情報を加算することで、現連続個数情報を更新し、

前記判断工程で、同じデータの連続数を計数中であり、且つ、現入力のデータが直前に入力されたデータと等しくない場合、又は、異なるデータの連続数を計数中であり、且つ、現入力のデータが直前に入力されたデータと等しい場合、計数中であった未出力のデータからデータ部、計数された連続数個数情報から連続数コード部を生成する工程とを備える。

【００１８】

【発明の実施の形態】

以下、添付図面に従って本発明に係る実施形態を説明する。

【００１９】

＜概要の説明＞

実施形態では、デジタル複写機を例にして説明することとする。

【００２０】

図１４は実施形態におけるデジタル複写機のブロック構成図である。図中、１は装置全体の制御を司る制御部であり、ＣＰＵ、その制御処理プログラムを記憶しているＲＯＭ、及びワークエリアとして使用するＲＡＭで構成される。２は原稿を読取るリーダ部であり、原稿画像をカラー画像として読取る撮像ユニット並びにＡＤＦ（Auto Document Feeder）を搭載している。３は前処理部であって各種補正処理（例えばシェーディング補正等が含まれる）を行う。４は読取って得た画像データを圧縮符号化する符号化部であり、例えばＪＰＥＧ符号化方法に従って圧縮符号化する。５は圧縮符号化データを一時的に記憶する記憶部である。従って、リーダ部２からは順次原稿画像を読み取ると、その圧縮符号化データが記憶部５に蓄積されていくことになる。６は復号部であって、記憶部５に蓄積された圧縮符号化データを復号する。７は画像処理部であって、復号部１２で復号して得られた各画素毎について記録色成分のデータの生成すると共に、後述する復号部１２からの各画素毎の情報に基づき、像域に特化した処理を行う。例えば、文字線画領域については、この結果は、プリンタエンジン８に出力され、印刷が行われることになる。プリンタエンジン８は、カラーレーザビームプリンタエンジンとするが、インク液を吐出するタイプでも勿論構わないので、印刷方式で限定されるものではない。

【００２１】

一方、前処理部３から出力された各色成分のデータは、像域判定部９に供給され、ここで画素単位にその如何なる像域にあるのかを判定し、１画素につきこの判定結果を１バイトの属性情報として出力する。

【００２２】

この像域判定部９で判定する内容としては、注目画素が網点中にある文字線画の画素であるか否か、有彩色／無彩色のいずれであるか、文字か否か（文字線画領域か中間調領域か）、文字線画である場合のその線分の太さ情報が含まれる。

【００２３】

文字線画か否かの判定は注目画素と隣接する画素との間の輝度変化が急峻であるか否か（閾値を越えるか）で判断する。そして、その急峻であると判定された画素が所定数連続する場合には文字線画と判定し、所定数に満たない場合には網点であると判断する。従って

、網点中にある文字か否かは、注目画素が文字であると判定され、そこから所定の距離範囲内に網点であると判定された画素数が複数存在するかで判断する。なお、文字線画でもなく、網点でもないと判断した場合には、中間調領域と判断すれば良いであろう。

【００２４】

また、有彩色か無彩色かの判断は、注目画素の各輝度色成分ＲＧＢの輝度値の差が閾値以下の場合に無彩色と判断する。更に精度良く判定するためには、注目画素を含む近傍に１つでも有彩色の画素が存在すれば、注目画素を有彩色と判断するようにする。

【００２５】

また、文字の太さ情報とは、文字であると判断された画素のエッジに沿う方向に対して直交する方向の文字線画と判定された画素数を計数すれば良いであろう。

【００２６】

上記の如く、像域判定部９は、１画素毎にその判定結果を８ビットのデータとして出力する。８ビットの内訳は、

Ｂｉｔ	意味	
7	予約（０）	
6	予約（０）	
5	文字太さ	
4	文字太さ	ビット４、５で文字・線画の太さ０～３を表現
3	予約（０）	
2	文字	文字線画＝１、非文字線画（中間調）＝０
1	網点中文字	網点中にある文字・線画＝１、それ以外＝０
0	無彩色	無彩色＝１、有彩色＝０

とした。

【００２７】

さて、像域判定部９では、上記の如く、１画素につき１バイトの像域情報が出力されてくるが、像域情報符号化部１０はこの情報をパックビッツ符号化を行うことで圧縮する。本実施形態における特徴部分は、この像域情報符号化部１０にあるので、その詳細について後述することとする。１１は像域情報符号化部１０でパックビッツ符号化された像域情報を一時的に記憶する記憶部である。１２は復号部であって、記憶部１１に記憶された圧縮符号化された像域情報を復号し、画像処理部７に供給する。従って、復号部６、１１は互いに同期して、復号結果を画像処理部７に出力することになる。

【００２８】

上記構成において、実施形態における像域情報符号化部１０の構成は、例えば図１５に示す構成となっている。

【００２９】

図中、１１は後述する監視部からの指示に従い、像域判定部９からの１バイト（画素の像域情報）の各ビットを条件つきでマスクするマスク部であり、１ページの初期状態では全てのビットを通過するようになっている。

【００３０】

１２はマスク部１１を介して入力した像域データを順次入力し、公知のパックビッツ符号化を行うパックビッツ符号化部である。１３は、記憶部１１に出力する際に、一時的に格納するためのバッファメモリである。１４はバッファメモリ１２に蓄積されるデータ量を監視する監視部（制御部１の制御下にある）であり、１枚の原稿画像に対応する属性情報の符号データの格納が完了する以前に予め設定された符号量（閾値）に達した場合に、マスク部１１でマスクするビットを決定すると共に、以下に説明するパックビッツ再符号化部１５に対する制御信号を生成する。１５はパックビッツ再符号化部であって、バッファメモリ１３に格納されている圧縮符号化された像域情報を再符号化し、再度、バッファメモリ１３に格納する。その際の再符号化条件（圧縮パラメータ）は、監視部１４から出力されてきた制御信号によって異なる（詳細は後述）。また、監視部１４は、この再符号化によるデータ量をも監視することになる。１６はパックビッツ再符号化部１５とバッフ

ァメモリ１３との間に位置し、パックビッツ再符号化部１５によるデータ読み込み及び書き込みを行うためのメモリ制御部である。

【００３１】

さて、上記の構成において、実施形態における監視部１４は、原稿１枚の画像を読込みを行う初期段階では、マスク部１１には、８ビットの全てのデータを通過するように設定する。そして、バッファメモリ１３に蓄積されていく像域情報の符号化データ量を監視し、１ページ分のデータが格納される以前に予め設定されたデータ量に達した（オーバーフロー状態という）と判断した場合、マスク部１１に対して所定のビットをマスクさせ、それ以降に入力される像域情報に制限を加える。また、既にバッファメモリ１３に蓄積さた符号化データ（像域情報）については、パックビッツ再符号化部１５で再符号化させる。この再符号化する際には、マスク部１１と同様、所定ビットをマスクするように設定する。

【００３２】

この結果、バッファメモリ１３には、オーバーフロー状態になったと判断した以降のデータについては、マスクされた結果に基づくパックビッツ符号化が行われることになり、オーバーフロー状態になったと判断したタイミングよりも前に符号化された像域情報の符号化データに対してパックビッツ再符号化部１５で再符号化することになる。監視部１４は、オーバーフロー状態になったとき、監視していたデータ量を一旦リセットし、パックビッツ符号化部１１から出力されてきたデータ量と、パックビッツ再符号化部１５での再符号化データ量との合計値が再びオーバーフローするか否かを監視続けることになる。そして、１ページの属性の圧縮符号化データ量が再びオーバーフローした場合には、更にマスクするビットを増やし、上記処理を繰り返すことになる。

【００３３】

実施形態におけるオーバーフロー（ＯＦ）の回数と、像域情報のマスク条件及びビットは、

ＯＦ回数	条件＆マスクビット
０	マスク無し
１	文字フラグ（ビット２）が０なら、無彩色（ビット０）を０
２	文字フラグが０ならば、文字太さ（ビット４、５）を０
３	文字フラグが０ななら、網点文字フラグ（ビット１）を０
４	文字フラグ以外全て０
５	全て０

とした。

【００３４】

上記内容を分かりやすく説明するのであれば、画素の属性をそれを包含する属性に変更することを意味する。例えば、最初のオーバーフローで無彩色を有彩色にするのは、有彩色空間内に無彩色空間が包含されると言えるからである。

【００３５】

なお、マスク部１１では、オーバーフロー回数が例えば２回目である場合には、１回目と２回目のＯＲ条件でマスクするビットは２つになる。一方、パックビッツ再符号化部１５では、上記の通りで良い。なぜなら、再符号化する対象は前回の符号化、或いは前回の再符号化で既にマスク済みであるからである。

【００３６】

また、上記の処理において、パックビッツ符号化部１２の処理そのものはオーバーフロー回数とは無縁の処理を単純に行い続ければ良い。一方、パックビッツ再符号化部１５は非常に高速な処理が望まれる。

【００３７】

本実施形態におけるパックビッツ再符号化部１５の再符号化する対象はパックビッツ符号化されたデータであることに着目し、その符号データのまま再符号化するようにした。以下、その具体的な解決策を説明することとする。

【００３８】

＜パックビッツ再符号化部１５の説明＞
図１は本実施形態におけるパックビッツ再符号化部のブロック図である。以下、同図のブロック図を用いて、その構成と処理内容を説明することとする。
【００３９】
図１において、１０１はデータ分割（分離）部であり、メモリ制御部１６を介して、ｅｎｃｏｄｅ＿ｄａｔａ（図中の１１１）から入力されるパックビッツ符号化データの長さ情報とデータ部を判別し、入力された長さ情報を解析することにより、長さ情報に続いて入力されるデータ部のデータ値がいくつ連続しているのか（以下、連続個数と記す）をデータ値と同時に出力する。すなわち、長さ情報とデータ部とを分離する、と言えば分かりやすい。
【００４０】
本実施形態のデータ分割部１０１はパックビッツ符号化データを受け取るencode_dataの他にencode_dataからパックビッツ符号化データが入力されていることを表す有効信号であるdata_valid（１１０）及び符号化データの入力が終了したことを表す終了信号data_end（１１３）、及び、データ分割部が符号化データを入力可をメモリ制御部１６に通知するレディ信号data_ready（１１２）を用いてパックビッツ符号化データを取得する。換言すれば、メモリ制御部１６は、このdata_ready信号がイネーブルとなると、次のデータをバッファメモリ１３から読出すことになる。
【００４１】
また、データ分割部１０１からの出力はdata（１１５）を用いてデータ値を、ｎｕｍ（１１６）を用いて連続個数の出力を行う。本実施形態では、前述のdata、numのバスの他に前記data及びnumから出力されている値が有効であることを表す有効信号valid（１１４）、受け側がデータの受信が可能であることを表すレディ信号ready（１１７）及びデータの出力が終了したことを表す終了信号end（１１８）を用いてデータの授受を行う。これら入出力信号の動作については後で説明する。
【００４２】
１０２はデータ処理部であり、監視部１４からの制御信号に従い、データ分割部１０１から出力されたデータ値に対して、先に説明した条件に従い、data（115）として入力されたデータ値に対してマスク（ビット値を０）にする処理を行い、masked_data（121）として出力する。
【００４３】
１０３はデータ結合部であり、データ処理部１０２がmasked_dataとして出力したデータ値及びデータ分割部１０１がnumから出力した連続個数を参照してデータ量を削減できる部分に対してパックビッツ方式による符号化処理を行って新たな符号化データを生成する。
【００４４】
尚、本実施形態ではnum、masked_dataの他にデータ分割部１０１で説明したvalid及びレディ信号ready、終了信号endを用いてデータ値及び連続個数を取得する。
【００４５】
データ結合部で生成した符号化データは、v_out（133）を用いて出力される。しかしデータ結合部１０３から出力される符号化データは通常のパックビッツ符号化データとは異なり、先にデータ部のデータ値が、データ部に続いて長さ情報がv_outから出力される。
【００４６】
そのため本実施形態のデータ結合部では、v_outから出力しているデータが長さ情報であるかデータ部であるかを判別するために、長さ情報有効信号であるl_valid及びデータ部有効信号であるd_vald、データの出力が終了したことを表す終了信号v_end（134）を用いて符号化データの出力を行う。
【００４７】
１０４はデータ出力部であり、データ結合部１０３から取得した符号化データを、通常のパックビッツ符号化データの順番になるように長さ情報とデータ部の並び順を入れ替えを

行う。上記データ結合部１０３及びデータ出力部１０４により、パックビッツ符号化データを再構成することになる。
本実施形態では並び換えたパックビッツ符号化データを８データ分をまとめて６４ビットとしてreencode_data（141）から出力すると共に、前記reencode_dataから出力したデータを格納する後段のメモリアドレスをaddress（142）から出力する。また、reencode_data及びaddressからデータ及びアドレスを出力する際には、前記データ及びアドレスが有効であることを表すように有効信号であるreencode_valdから１を出力する。
【００４８】
また、データ出力部からのアドレス及びデータの出力が終了すると、reencode_end（143）から１を出力することでデータの出力が終了したことを後段に知らせる。
【００４９】
データ分割部１０１の動作を手順として示すと、図２に示すフローチャートのようになる。
【００５０】
データ分割部１０１は動作を開始すると内部カウンタ（図２ではｉと表記）を初期化する。具体的にはカウンタ ｉ を０に設定し（ステップＳ２０１）、次にパックビッツ符号化データの入力が終了しているかどうかの判断を行う（ステップＳ２０２）。
【００５１】
データの入力終了の判断は先に説明したように「長さ情報として128（16進数表記で80h、以下数字に続く括弧内は１６進数表記を表す）を入力する」という方法もあるが、本実施形態ではdata_endから「TRUE＝１」を入力することによりデータの入力終了を表すこととする。
【００５２】
ステップＳ２０２においてデータ入力が終了していない（data_end = 0）と判断されると、データ分割部はencode_dataからデータが入力されるのを待つ。ここで、従来の技術で示した（３）及び（４）のようにパックビッツ符号化データは最初に長さ情報が入力されるため、具体的には長さ情報が入力されるのを待つことになる。
【００５３】
encode_dataから最初の符号化データである長さ情報（図２ではＬと表す）が入力されると（ステップＳ２０３）、データ分割部では入力されたＬの値を参照して長さ情報が「同じデータが連続する数」を表しているか「異なるデータが連続する数」を表しているのかを判断する。具体的には、入力された長さ情報Ｌを正負の符号無しデータとし、閾値「１２８（80h）」と比較し（ステップＳ２０４）、Ｌが１２８より大きい場合は「同じデータが連続する数」、１２８未満の場合は「異なるデータが連続する数」と判断する。
【００５４】
入力された長さ情報Ｌが128（80h）より大きい場合、データ分割部では長さ情報Ｌから同じデータの連続する数を求めて変数Ｎに代入する。本実施形態では同じデータが連続する数は２の補数により負の値として表現されているため、長さ情報Ｌから連続数の算出を行う（ステップＳ２０５）。具体的には
Ｎ ＝ ２５６ － Ｌ ＋１
により連続数を求めてＮに代入する。
【００５５】
同じデータが連続する数を算出すると、データ分割部は再びencode_dataからパックビッツ符号化データが入力されるのを待つ。パックビッツ符号化データでは長さ情報が128以上の場合、次のデータは連続するデータの値が入力されるため、ここではデータ部（データ値）の入力を待つ。
【００５６】
encode_dataからデータ部（図２ではＤと表す）が入力されると（ステップＳ２０６）、データ分割部はnum出力（１１６）からデータ連続数であるＮを、data出力（１１５）からデータ値であるDを出力する（ステップＳ２０７）。

【００５７】
これにより、「データ値ＤがＮ個ある」という情報を後段に送り、再びステップＳ２０２によるデータ入力終了の判断に戻る。
【００５８】
また、入力された長さ情報Ｌが128（80h）未満の場合、データ分割部では長さ情報Ｌから異なるデータが連続する数を求めて変数Ｎ’に代入する（ステップＳ２０８）。本実施形態では異なるデータが連続する数は長さ情報に１を加えた値であるため、具体的には
Ｎ’＝Ｌ　＋　１
により異なるデータの数を求める。
【００５９】
異なるデータの数を算出すると、データ分割部は再びencode_dataからパックビッツ符号化データが入力されるのを待つ。パックビッツ符号化データでは長さ情報が128以下の場合、長さ情報に続いて先に算出したＮ’の数だけデータの値が入力される。従ってここではデータ部（データ値）の入力を待つ。
【００６０】
encode_dataからデータ部Dが入力されると（ステップＳ２０９）、データ分割部はnum出力（１１６）から１を、data出力（１１５）からデータ値Ｄを出力する（ステップＳ２１０）。これにより、「dataから出力したデータ値Ｄは１つである」という情報を後段に送る。
【００６１】
dataからデータ値Ｄを出力すると、データ分割部は内部カウンタ　ｉ　に１を加算し（ステップＳ２１２）、Ｎ'との比較を行う（ステップＳ２１２）。カウンタ値　ｉ　がＮ'より小さい場合は、引き続きデータ値が入力されるためステップＳ２０９に戻りencode_dataからデータ部が入力されるのを待つ。
【００６２】
またステップＳ２１２により比較した結果カウンタ値ｉとＮ'の値が等しい場合は、データ値がＮ'の個数入力されたため、カウンタｉを０に初期化して（ステップＳ２１３）、ステップ２０２によるデータ入力終了の判断に戻る。
【００６３】
上記ステップＳ２０２～２１３の動作を繰り返すことでパックビッツ符号化データの入力を行い、最後にステップ２０２においてdata_end=1が入力されてパックビッツ符号化データの入力終了が検出されると、データ分割部１０１はデータ出力を終了したことを表すためにend出力から１を出力して動作を終了する。
【００６４】
図１０にデータ分割部１０１の入出力タイミングチャートの一例を示す。
【００６５】
図１０において、ｃｌｋはクロックを表し、各信号はｃｌｋに同期して入出力が行われる。data_valid、encode_data、data_ready及びdata_endの各信号はパックビッツ符号化データをデータ分割部に入力するための信号である。N、N'及びｉはデータ分割部内で使用する変数及びカウンタ値を表し、valid、num、data、ready及びend信号はデータ部及び連続個数をデータ処理部１０２及びデータ結合部１０３に出力するための信号である。
【００６６】
尚、図１０ではencode_dataから入力されるパックビッツ符号化データのうち、長さ情報を表す部分には二重下線を引いて表示している。
【００６７】
図１０においてデータ分割部１０１の動作が開始すると、図１０のタイミングｔ０において内部カウンタｉの初期化動作が行われる（ステップＳ２０１）。その後、ｔ１からdata_valid=1とすることによりencode_dataからデータ分割部にパックビッツ符号化データの入力が開始される。
【００６８】

最初にデータ分割部に入力されるバックビッツ符号化データは長さ情報であるため、データ分割部はｔ１において入力された値２５３(＝FDh)を長さ情報Ｌとして取得する（ステップＳ２０３）。その後取得した長さ情報Ｌと１２８（80h）を比較し（ステップＳ２０４）、

Ｌ ＜ １２８

の条件が満たされないため、取得した長さ情報Ｌを用いて、ステップ２０５に示した演算によりデータ連続数４（04h）を算出してＮに格納する。

【００６９】

その後、タイミングｔ２においてencode_dataから入力された値０（00h）をデータ値Ｄとして取得し（ステップＳ２０６）、タイミングｔ３においてｎｕｍからデータ連続個数Ｎを、dataからデータ値Dを出力した後（ステップＳ２０７）、ステップＳ２０２においてdata_endを参照してデータ入力が終了しているかどうかを判断する。

【００７０】

タイミングｔ３ではdata_endから１が入力されていないため、ステップＳ２０３に進み、ｔ３においてencode_dataから入力された値248(F8h)を長さ情報Ｌとして取得する（ステップＳ２０３）。そして長さ情報Ｌと128(80h)を比較し、ステップＳ２０５に進んでデータ連続数９(09h)を算出してＮに格納する。そしてタイミングｔ４でencode_dataからデータ値Dとして１(01h)を取得して（ステップＳ２０６）、タイミングｔ４においてｎｕｍからデータ連続個数Ｎを、dataからデータ値Ｄを出力する。

【００７１】

その後再びステップＳ２０２においてデータ入力終了の判断を行い、タイミングｔ５におけるdata_endは０であるのでステップＳ２０３に進み、タイミングｔ５においてencode_dataから入力されている６(06h)を長さ情報Ｌとして取得する。

【００７２】

ステップＳ２０４により長さ情報Ｌと１２８(80h)を比較し、

Ｌ ＜ １２８

の条件を満たすため、ステップ２０８に示した式により長さ情報Ｌから長さ情報に続いて入力されるデータ値の個数７（07h）を算出してＮ’に格納（ステップＳ２０８）した後、eocnde_dataからデータ部Ｄの入力を待つ。

【００７３】

タイミングｔ６においてencode_dataからデータ値Ｄとして「２(02h)」が入力されるとすると（ステップ２０９）、ｔ７でnumからデータ連続数として１（01h）を、dataからデータ値Ｄである「２(0h)」を出力する（ステップＳ２１０）。出力値が「２」のまま変らないのは、第１回めのオーバーフロー時のマスク条件が、文字であるビット２が０である場合には、無彩色を示すビット０を０、すなわち、有彩色にするからである。そして内部カウンタｉに１を加算して（ステップＳ２１１）、ステップ２０８で算出したＮ'と比較する（ステップＳ２１２）。

【００７４】

タイミングｔ７では内部カウンタｉの値は１であるためＮ’に格納したデータ入力個数である７と等しくないため、ステップＳ２０９に戻り再びencode_dataからデータ部Ｄが入力されるのを待つ。

【００７５】

タイミングｔ７ではdata_validが１であるためencode_dataから入力されているデータは有効であるが、後述するデータ分割部１０３がデータの入力準備ができていないためready信号を０にしている。そのため、データ分割部でもdata_readyから０を出力してデータの取得を停止する。この時、encode_dataから入力されているデータは次のｃｌｋまで保持される。

【００７６】

ｔ８になりデータ分割部１０３がデータの入力が可能になりreadyから１が入力されると、データ分割部もdata_readyから１を出力してencode_dataからデータ値Ｄとして３（03h

）が取得される。

【００７７】

以後、ステップＳ２０９～ステップＳ２１２により同様の動作をＮ’の回数だけ繰り返し、encode_dataから入力された値としてタイミングｔ９で２（02h）が出力される。理由は、文字であることを示すビット２が０であるので、ビット０がマスクされるからである。

【００７８】

続く、ｔ１０で２（02h）、ｔ１１で３(03h)、ｔ１３で４(04h)、ｔ１４で３（03h）、ｔ１５で１(01h)を取得して、dataとしてタイミングｔ10で２（02h）、ｔ１１で２（02h）、ｔ１２で４(04h)、ｔ１４で２(02h)、ｔ１５で０（00h）を出力することになる。これらはマスク条件に従って行われるのは先に説明した通りである。

【００７９】

タイミングｔ１５では内部カウンタ ｉ の値が７となりデータ入力個数Ｎ’と等しくなるため、ステップＳ２１２における比較の後、内部カウンタｉを０に初期化し（ステップＳ２１３）、ステップＳ２０２に戻りdata_endを参照してデータ入力が終了しているかどうかを判断する。

【００８０】

タイミングｔ１５ではdata_endから１が入力されていないため、ステップＳ２０３に進み、タイミングｔ１５においてencode_dataから入力された値253(FDh)を長さ情報Ｌとして取得し（ステップＳ２０３）、ステップＳ２０４で128(80h)を比較した後、ステップＳ２０５に進んでデータ連続数４(04h)を算出してNに格納する。そしてタイミングｔ１６でencode_dataからデータ値Ｄとして０(00h)を取得して（ステップＳ２０６）、タイミングｔ１７においてｎｕｍからデータ連続個数Ｎを、dataからデータ値Ｄを出力し、ステップＳ２０２に戻る。

【００８１】

タイミングｔ１７ではencode_endから１が入力されているため、データ分割部ではステップＳ２０２においてデータ入力が終了したと判断し、タイミングｔ１８においてend出力から１を出力することによりデータ出力の終了を後段に知らせることにより（ステップＳ２１４）、本動作を終了する。

【００８２】

図１０のmasked_dataはデータ処理部１０２からの出力であり、本実施形態ではデータ処理部１０２においてデータ分割部１０１のdataから出力された値に対して、最初のオーバーフローの例を示した。すなわち、文字であることを示すビット２が０の場合、ビット０を０にするものである。しかし、次のオーバーフロー（２回めのオーバーフロー）の場合には、条件がまた異なることになることは明らかであろう。

【００８３】

尚、本実施形態のデータ処理部１０２では図１０に示す様にデータ処理部にdataから値が入力されてからｃｌｋ遅延無くmasked_dataを出力しているため、masked_dataから出力されたデータと、データ分割部のnumから出力されているデータの連続個数を表す情報は同じタイミングでデータ結合部に与えられる。

【００８４】

次に、実施形態におけるデータ結合部１０３の動作処理を図３乃至図８のフローチャートに従って説明する。

【００８５】

データ連結部１０３がその処理を開始すると、num及びmasked_dataから最初のデータ連続個数及びデータ処理部１０２で処理されたデータ値の入力を待つ。

【００８６】

データ連続個数及びデータ値が入力されると（ステップＳ３０１）、その値を初期値としてnumから入力されたデータ連続個数を内部変数ｃｎｔに、masked_dataから入力されたデータ値を内部変数data_bufに格納する（ステップＳ３０２）。

【００８７】

そしてcntに格納した値を参照し（ステップＳ３０３）、その値が１以外である場合はmasked_dataから入力されたデータ値が「同じデータ値が連続している」部分のデータ値であると判断し、その状態をデータ連続部で保持するために内部変数continueを１に設定する（ステップＳ３０４）。

【００８８】

また、ｃntに格納した値が１である場合は、masked_dataから入力されたデータ値が「同じデータが連続していない」部分のデータ値であると判断し、この状態をデータ連続部で保持するために内部変数continueを０に設定する（ステップＳ３０５）。

【００８９】

以上により、次のデータを入力するための準備が終了する。

【００９０】

次にデータ結合部はデータ入力の終了を判断する（ステップＳ３０６）。本実施形態ではデータ分割部がend(118)が「１＝high」にすることでデータの入力が終了したと判断し、図８の処理へと分岐する。end(118)から１が入力されていない場合には、データ分割部はnum(116)とmasked_data（121）から次のデータ連続数及びデータ値が入力されるのを待つ。

【００９１】

データ連続数とデータ値が入力されると（ステップＳ３０７）、データ結合部は内部変数continueの値により分岐を行う（ステップＳ３０８）。continueの値が「同じデータが連続していない」部分を表す０である場合、さらにmasked_dataから入力されたデータ値と内部変数data_bufの値を比較する（ステップＳ３０９）。ここでmasked_dataから入力されたデータ値とdata_bufの値が異なる場合は図４の処理へ分岐する。また、masked_dataから入力されたデータ値とdata_bufの値が等しい場合は、図５の処理へ分岐して「同じデータが連続していない部分」を終了するための処理と、新たに「同じデータが連続している部分」が始まるための処理を行う。

【００９２】

一方、ステップＳ３０８により内部変数continueの値が「同じデータが連続している」部分を表す１であった場合、同様にmasked_dataから入力されたデータ値と内部変数data_bufの値を比較し（ステップＳ３１０）、masked_dataから入力されたデータ値とdata_bufの値が異なる場合は図６の処理へ分岐する。また、masked_dataから入力されたデータ値とdata_bufの値が等しい場合は、図７へ分岐して「同じデータが連続している部分」を続けるための処理を行う。

【００９３】

図４はデータ結合部における、内部変数continueが０かつmasked_dataから入力されたデータ値と内部変数data_bufの値が異なる場合（図３におけるステップＳ３０９がｙｅｓ）の処理を表すフローチャートである。

【００９４】

この場合、新たなパックビッツ符号化データを生成するためにdata_bufに格納されている値が必要であるためdata_bufに格納されている値をv_outから出力する（ステップＳ４０１）。そしてnumから入力されているデータ連続数を参照する（ステップＳ４０２）。

【００９５】

データ連続数が１以外であれば、それはmasked_dataから入力されている値が１つ以上連続している事を表しているため、「同じデータが連続していない」部分はここで終了する。そのため内部変数continueの値を「同じデータが連続している」部分を表す１に設定し（ステップＳ４０３）、それまで続いていた「異なるデータが連続している」部分のデータ数が格納されている内部カウンタcntから１を減ずることで「異なるデータが連続する数」を表す長さ情報を求め、その長さ情報をv_outから出力する（ステップＳ４０４）。

【００９６】

その後、内部変数cntを０に初期化して（ステップＳ４０５）から、numから入力されたデータ連続数をｃｎｔに加算し（ステップＳ４０６）、masked_dataから入力されたデータ

値を内部変数data_bufに格納し（ステップＳ４０７）、次のデータ受け取り準備を行った後で、図３のステップＳ３０６に戻る。

【００９７】

また、ステップＳ４０２においてｎｕｍから入力されたデータ連続数が１であった場合は、引き続き「同じデータが連続していない」部分であることを表している。この場合はまず内部変数ｃｎｔの値を参照し、１２８（80h）であるかを判断する（ステップＳ４０８）。本実施形態では８ビットでパックビッツ符号データを表しているため、「同じデータが連続している」部分及び「同じデータが連続していない」部分の数は128までしか表す事が出来ない。そのため内部ｃｎｔの値が128の場合、「同じデータが連続していない部分」が128回続いているという情報を出力する必要が生ずる。そのためステップＳ４０４により長さ情報を算出して（この場合「異なるデータが128個連続しているので」長さ情報として127（7Fh）が生成される）v_outから出力する。

【００９８】

その後、内部変数cntを０に初期化して（ステップＳ４０５）から、numから入力されたデータ連続数（ここでは１）をｃｎｔに加算し（ステップＳ４０６）、masked_dataから入力されたデータ値を内部変数data_bufに格納し（ステップＳ４０７7）、次のデータ受け取り準備を行った後で図３のステップＳ３０６に戻る。

【００９９】

ステップＳ４０８において内部変数cntの値が128でない（128未満）である場合は、numから入力されたデータ連続数（ここでは１）をｃｎｔに加算し（ステップＳ４０６）、masked_dataから入力されたデータ値を内部変数data_bufに格納して（ステップＳ４０７）、次のデータ受け取り準備を行った後で、図３のステップＳ３０６に戻る。

【０１００】

図５はデータ結合部における、内部変数continueが０かつmasked_dataから入力されたデータ値と内部変数data_bufの値が同じ場合（図３におけるステップＳ３０９でＮｏ）の処理を表すフローチャートである。

【０１０１】

この場合、一つ前の処理まではcontinueが０であるため「同じデータが連続しない」部分の処理を行っていたが、masked_dataから入力されているデータ値と内部変数data_bufの値が同じであるため「同じデータが連続しない」部分は一つ前のデータで終了していたことになる。そこで内部変数continueの値を「同じデータが連続している」部分を表す１に設定し（ステップＳ５０１）、内部変数cntの値を参照する（ステップＳ５０２）。cntの値が１でなければ、それまで続いていた「異なるデータが連続している」部分のデータ数を算出して出力する必要がある。

【０１０２】

そのためには、現在data_bufに格納されている値は「同じデータが連続している」部分に含まれるため、実際に「同じデータが連続していない」部分のデータ数は内部カウンタcntから１を減した値となる。また、「異なるデータが連続する数」を表す長さ情報は「異なるデータが連続している」数から１を減じた値であるため、ここでは内部変数cntから２を減じて長さ情報を求め、その長さ情報をv_outから出力する（ステップＳ５０３）。

【０１０３】

そしてmasked_dataから入力されるデータ値と同じデータがdata_bufに入っているため内部変数ｃｎｔを１に設定（ステップＳ５０４）してから、numより入力されたデータ連続数をｃｎｔに加算（ステップＳ５０５）して次のデータ受け取り準備を行い、図３におけるステップＳ３０６に戻る。

【０１０４】

また、ステップＳ５０２においてcnt値が１であった場合は、cntの値にnumから入力されたデータ連続数を加算する（ステップＳ５０６）。

【０１０５】

本実施形態では８ビットでパックビッツ符号データを表しているため、「同じデータが

連続している」部分の数は128までしか表す事が出来ないため、内部ｃｎｔの値が129以上の場合、「同じデータが連続している」部分が128回続いているという情報を出力する必要が生ずる。そのため内部変数cntの値を参照し（ステップＳ５０７）、その値が１２９以上である場合はdata_buf　に格納されているデータ値をv_outから出力し（ステップＳ５０８）、それに続いて「同じデータが128個連続している」ことを表す長さ情報である129(81ｈ)をv_outから出力する（ステップＳ５０９）。尚、この129という長さ情報は、従来技術で説明した長さ情報とコードをテーブルとして記憶保持し、それを参考して求めてもいいし、同じデータがｎ個続く事を表す長さ情報が

２５６－ｎ＋１

であることから求めてもよい。ステップＳ５０８により長さ情報が出力されると、データ結合部は内部変数cntから後段にパックビッツ符号データとして出力した分の「１２８」を減じ（ステップＳ５１０）、次のデータ受け取り準備を行った後で、図３のステップＳ３０６に戻る。

【０１０６】

また、ステップＳ５０７において内部cntの値が１２９未満である場合は、図３のステップＳ３０６に戻ることになる。

【０１０７】

尚、ここではパックビッツ符号化に必要なデータ値は既にdata_bufに格納されているのでmasked_dataから入力されたデータ値を内部変数data_bufに格納する必要は無いが、格納してもよい。

【０１０８】

図６はデータ結合部における、内部変数continueが１かつmasked_dataから入力されたデータ値と内部変数data_bufの値が異なる場合（図３におけるステップＳ３１０がＮｏ）の処理を表すフローチャートである。

【０１０９】

この場合masked_dataから入力されるデータ値をdta_bufの値が異なるため、「同じデータが連続している」部分はここで終了する。そのためdata_bufに格納されている値をv_outから出力する（ステップＳ６０１）。そしてnumから入力されているデータ連続数を参照する（ステップＳ６０２）。

【０１１０】

データ連続数が１以外であれば、それはmasked_dataから入力されている値が１つ以上連続している事を表しているため、再び「同じデータが連続している」部分が開始される。そのため内部変数continueの値は「同じデータが連続している」部分を表す１のままにしておき、それまで続いていた「同じデータが連続している」部分のデータ数が格納されている内部カウンタcntから長さ情報を求め（256からcntの値を減じ、さらに１を加えることにより求める）、その長さ情報をv_outから出力する（ステップＳ６０３）。

【０１１１】

その後、内部変数cntを０に初期化して（ステップＳ６０４）から、masked_dataから新たなデータ値を取得してdata_bufに格納し（ステップＳ６０５）、さらにnumから入力されたデータ連続数をｃｎｔに加算する（ステップＳ６０６）。こうして次のデータ受け取り準備が終了すると、図３のステップＳ３０６に戻ることになる。

【０１１２】

また、ステップＳ６０２においてｎｕｍから入力されたデータ連続数が１であった場合は「同じデータが連続していない」部分が始まる事を表している。この場合はまず内部変数continueの値を「同じデータが連続していない」部分を表す0に設定して（ステップＳ６０７）、内部変数cntの値を参照する（ステップＳ６０８）。

【０１１３】

ｃntの値が１以外であれば、「同じデータが連続している」部分のデータ数が格納されている内部カウンタcntから長さ情報を求め（256からcntの値を減じ、さらに１を加えることにより求める）、その長さ情報をv_outから出力する（ステップＳ６０３）。

【０１１４】
その後、内部変数cntを０に初期化して（ステップＳ６０４）から、masked_dataから新たなデータ値を取得してdata_bufに格納し（ステップＳ）、さらにnumから入力されたデータ連続数（ここでは１）をｃｎｔに加算する（ステップＳ６０６）。こうして次のデータ受け取り準備が終了すると、図３のステップＳ３０６に戻り、データ入力終了の判断を行うことになる。
【０１１５】
また、ステップＳ６０８により内部変数cntの値が１であった場合は、ステップＳ６０１で出力したデータ値を「同じデータが連続しない」部分の最初のデータとして処理を行う。そのためmasked_dataから新たなデータ値を取得してdata_bufに格納し（ステップＳ６０５）、さらにnumから入力されたデータ連続数（ここでは１）をｃｎｔに加算する（ステップＳ６０６）。こうして次のデータ受け取り準備が終了すると、図３のステップＳ３０６に戻る。
【０１１６】
図７はデータ結合部における、内部変数continueが１かつmasked_dataから入力されたデータ値と内部変数data_bufの値が同じ場合（図３におけるステップＳ３１０がｙｅｓ）の処理を表すフローチャートである。
【０１１７】
この場合masked_dataから入力されるデータ値をdta_bufの値が同じため、「同じデータが連続している」部分はさらに続く。そのためデータが連続する数を求めるため内部変数cntにnumから入力されたデータ連続数を加算する（ステップＳ７０１）。
【０１１８】
また、本実施形態では８ビットでパックビッツ符号データを表しているため、「同じデータが連続している」部分及び「同じデータが連続していない」部分の数は図５の説明で述べた様に「１２８」までしか表す事が出来ない。そのため内部ｃｎｔの値が１２９以上の場合、「同じデータが連続している」部分が１２８回続いているという情報を出力する必要が生ずる。そのため内部変数cntの値を参照し（ステップＳ７０２）、その値が１２９以上である場合はdata_bufに格納されているデータ値をv_outから出力し（ステップＳ７０３）、それに続いて「同じデータが128個連続している」ことを表す長さ情報である「129(81h)」をv_outから出力する（ステップＳ７０４）。尚、この「１２９」という長さ情報は従来技術で説明したテーブルを保持するようにし、それを参考して求めてもいいし、同じデータがｎ個続く事を表す長さ情報が
２５６－ｎ＋１
であることから求めてもよい。ステップＳ７０４により長さ情報が出力出力されると、データ結合部は内部変数cntから後段にパックビッツ符号データとして出力した分の「１２８」を減じ（ステップＳ７０５）、次のデータ受け取り準備を行った後で、図３におけるステップＳ３０６に戻る。
【０１１９】
また、ステップＳ７０２により内部変数cntが128以下の場合は、パックビッツ符号化に必要なデータは生成されないため、そのまま図３のステップＳ３０６に戻り、データ入力終了の判断を行うことになる。
【０１２０】
尚、図７のフローチャートではパックビッツ符号化に必要なデータ値は既にdata_bufに格納されているのでmasked_dataから入力されたデータ値を内部変数data_bufに格納する必要は無いが、格納してもよい。
【０１２１】
図８は図３のステップＳ３０６によりend(118)から１が入力され、データの入力が終了したと判断した場合のフローチャートを示す。
【０１２２】
図８ではまず始めに内部変数data_bufに格納しているデータ値をv_outから出力し（ステ

ップＳ８０１）、その後内部変数continueの値を参照する（ステップＳ８０２）。
【０１２３】
ｃontinueの値が０の場合、ステップＳ８０１で出力したデータ値は「データが連続していない」部分の最後のデータ値であるため、内部変数cntから「異なるデータが連続する」部分の長さ情報を生成して出力する。具体的には内部変数cntの値から１を減じてv_outから出力する（ステップＳ８０３）。
【０１２４】
そして後段にデータの出力終了を知らせるためにv_end（134）に１を出力して（ステップＳ８０４）動作を終了する。
【０１２５】
一方ステップＳ８０２においてcontinueが１であった場合は、ステップＳ８０１で出力したデータ値は「データが連続している」部分のデータ値であるため、内部変数cntから「同じデータが連続する」部分の長さ情報を生成して出力する。具体的には内部変数cntの値をもちいて
２５６－ｃｎｔ＋１
の値を求め、v_outから出力する（ステップＳ８０５）。
【０１２６】
そして後段にデータの出力終了を知らせるためにv_end（134）に１を出力して（ステップＳ８０４）動作を終了する。
【０１２７】
尚、本実施形態のデータ結合部103では、図３乃至図８において内部変数data_bufに格納したデータ値をv_outから出力する場合は同時にd_valid（131）から１を出力し、また内部変数cntから生成した長さ情報をv_outから出力する場合は同時にl_valid（132）から１を出力する。これによりv_out（133）から出力されるデータの有効信号、及びv_outから出力したデータがパックビッツ符号化データのデータ部であるか長さ情報であるかの区別を後段に知らせている。
【０１２８】
また、図３乃至図８において、図６のステップＳ６０３や図７のステップＳ７０４の様に、num及びdataからデータ連続数及びデータ値の取得を行う事でv_outからデータ部および長さ情報の二つのデータを出力しなければならない場合は、v_outからの出力が終了するまでready（117）を０（すなわち、次のデータの入力要求をしない）にすることによりデータ結合部１０１及び画像処理部１０２からのデータ連続数及びデータ入力を停止させる必要がある。
【０１２９】
図１１は、実施形態におけるデータ結合部１０３の入出力タイミングチャートの一例を示している。
【０１３０】
図１１において、ｃｌｋはクロックを表し、各信号はｃｌｋに同期して入出力が行われる。valid、num、masked_data、ready及びend信号はデータ分割部で生成されたデータ連続数及びデータ処理部１０２で生成されたデータ値をデータ結合部１０３に入力するための信号である。
【０１３１】
ｃｎｔ、data_buf及びcontinueはデータ結合部内で使用する内部変数を表し、d_valid、l_valid、v_out及びv_end信号はパックビッツ符号化データのデータ部および長さ情報をデータ出力部１０４に出力するための信号である。
【０１３２】
尚、図１１ではv_outから出力されるパックビッツ符号化データのうち、長さ情報を表す部分には二重下線を引いて表示している。
【０１３３】
図１１においてタイミングｔ３においてnum及びmasked_dataから最初のデータ連続数及び

データ値が入力される（ステップＳ３０１）と、データ結合部１０３ではｎｕｍから入力された4（04h）を内部変数ｃｎｔへ、masked_dataから入力された０（00h）を内部変数data_bufに格納する（ステップＳ３０２）。

【０１３４】

そしてステップ３０３によりcntの値を参照し、値が１ではないのでステップ３０４により内部変数continueを１に設定する。

【０１３５】

その後ステップ３０６によりデータ入力終了の判断を行い、endから１が入力されていないので再びnum及びmasked_dataからデータ連続数及びデータ値が入力されるのを待つ。

【０１３６】

タイミングｔ５においてnumから9（09h）、masked_dataから０（00h）が入力されると（ステップＳ３０７）、データ結合部１０３はステップＳ３０８により内部変数continueの値を参照し、その値が１であるのでステップ３１０に進みmasked_dataから入力されたデータ値とdata_bufに格納されている値の比較を行う。タイミングｔ３ではmasked_dataのデータ値とdata_bufの値がともに０（00h）で等しいため、データ結合部の動作は図７のステップＳ７０１へ進む。

【０１３７】

ステップ７０１により内部変数cntの値とｎｕｍから入力されたデータ連続数を加算し、加算後のcntの値を129と比較する（ステップＳ７０２）。加算した結果、内部変数cntの値は図１１のタイミングｔ６に示す様に１３（0Dh）であり、129よりも小さいため図３のステップ３０６に戻る。

【０１３８】

タイミングｔ６において再びステップ３０６によりデータ入力終了の判断を行い、endから１が入力されていないのでnum及びmasked_dataからデータ連続数及びデータ値が入力されるのを待つ。

【０１３９】

タイミングｔ７においてnumから１（01h）、masked_dataから２（02h）が入力されると（ステップＳ３０７）、データ結合部１０３はステップＳ３０８により内部変数continueの値を参照し、その値が１であるのでステップＳ３１０に進みmasked_dataから入力されたデータ値とdata_bufに格納されている値の比較を行う。タイミングｔ７ではmasked_dataのデータ値とdata_bufの値が異なるため、データ結合部の動作は図６のステップＳ６０１へ進む。

【０１４０】

ステップＳ６０１によりdata_bufに格納した値をd_validを１にすると同時にv_outから出力した後、numから入力された値を参照し（ステップＳ６０２）、その値が１であるためステップＳ６０７により内部変数continueを０に設定して「同じデータが連続しない」部分が始まる準備を行う。

【０１４１】

また、ステップＳ６０８により内部変数cntの値を参照し、１（01h）でないためステップＳ６０３により長さ情報

２５６－１３＋１

を算出してl_validを１にすると同時にv_outから244(F4h)を出力する。そして内部変数cntを０に初期化し（ステップＳ６０４）、masked_dataから入力されているデータ値である２（02h）をdata_bufに格納（ステップＳ６０５）、さらにnumから入力されているデータ連続数「１（01h）」をcntに加算して、cntの値を１（01h）にして図３のステップＳ３０６に戻る。

【０１４２】

尚、上記説明した動作はそれぞれ、ステップＳ６０１のdata_bufの値の出力をタイミングｔ７、ステップ６０７による内部変数continueに０を設定する動作をタイミングｔ７、ステップＳ６０３の長さ情報の出力をタイミングｔ８、ステップＳ６０５及びステップＳ６

０６におけるデータ値の取得及びデータ連続数の加算をタイミングｔ８で行うことになる。

【０１４３】

また、タイミングｔ７で入力されたｎｕｍ及びmasked_dataからデータ連続数及びデータ値については、データ結合部のv_outからデータ部および長さ情報の二つのデータを出力する必要があるため、タイミングｔ７においてデータ結合部のreadyから１を出力し、タイミングｔ７で入力されたｎｕｍ及びmasked_dataから入力された値をタイミングｔ８まで保持して、タイミングｔ８においてnum及びmasked_dataから必要な値の取得を行っている。

【０１４４】

次のデータとしてタイミングｔ９でｎｕｍから１（01h）、masked_dataから２（02h）が入力されると、データ結合部はステップＳ３０８によりcontinue値を参照する。タイミングｔ8におけるcontinue値は０であるため、ステップＳ３０９によりmasked_dataから入力された値２と、data_bufの値を比較する。タイミングｔ８におけるdata_bufの値は２であるためデータ結合部の動作は図５のステップＳ５０１へ進むことになる。

【０１４５】

ステップＳ５０１により内部変数continueの値を「同じデータ連続する」部分を表す１に設定し、ステップＳ５０２により内部変数cntの値を参照する。このときのcntの値は１であるためステップＳ５０６に進んでｃｎｔの値とｎｕｍから入力されるデータ連続数の加算を行う。タイミングｔ９におけるnumの値は１（01h）であるためcntの値は２（02h）となる。

【０１４６】

次にステップＳ５０７においてcntの値を参照するが、この時のcntの値は上に示したように２であり、129未満であるためにそのまま図３のステップＳ３０６に戻る。

【０１４７】

タイミングｔ１０及びタイミングｔ１１における動作は、入力される値及び動作がタイミングｔ９の場合と同じであるためここでは説明を省略するが、タイミングｔ１０及びタイミングｔ１１の動作により内部変数cntの値はそれぞれタイミングｔ１１では３（03h）、タイミングｔ１２では４（04h）となり、continueの値は１のままである。

【０１４８】

タイミングｔ１２においてnumから1（01h）、masked_dataから４（04h）が入力されると、データ結合部はステップＳ３０７からＳ３０８へ進み、内部変数continueが１であるためステップＳ３１０へ進む。そこでmasked_dataから入力された値である４と、data_bufに格納されている値である２を比較し、値が異なっているので図６のステップＳ６０１に進む。

【０１４９】

ステップＳ６０１によりv_outからdata_bufの値である２（02h）を出力し、numから入力されているデータ連続数が１であるためにステップＳ６０２からステップ６０７に進む。ここで「同じデータが連続連続しない」ことを表すようにcontinueに０を設定し、ステップＳ６０８によりcntの値を参照する。タイミングｔ１２におけるcntの値は４（04h）であるため、ステップＳ６０３に進んでcntの値から「同じデータが連続する」数を示す長さ情報を算出してv_outから出力し、ステップＳ６０４においてcntを初期化した後、masked_dataのデータ値である4（04h）をdata_bufに格納し、初期化後のcntにnumから入力されたデータ連続数1（01h）を加算して図３のステップＳ３０６に戻る。

【０１５０】

尚、上記説明した動作はそれぞれ、ステップＳ６０１のdata_bufの値の出力がタイミングｔ１２、ステップＳ６０７による内部変数continueに０を設定する動作をタイミングｔ１２、ステップＳ６０３の長さ情報の出力がタイミングｔ１３、ステップＳ６０５及びステップＳ６０６におけるデータ値の取得及びデータ連続数の加算はタイミングｔ１３で行われる。

【０１５１】
また、タイミングｔ１２で入力されたｎｕｍ及びmasked_dataからデータ連続数及びデータ値については、データ結合部のv_outからデータ部および長さ情報の二つのデータを出力する必要があるため、タイミングｔ１２においてデータ結合部のreadyから１を出力し、タイミングｔ１２で入力されたｎｕｍ及びmasked_dataから入力された値をタイミングｔ１３まで保持して、タイミングｔ１３においてnum及びmasked_dataから必要な値の取得を行っている。
【０１５２】
タイミングｔ１４においてnumから1（01h）、masked_dataから２（02h）が入力されると、データ結合部はステップＳ３０７からＳ３０８へと処理を進め、内部変数continueが０であるためステップＳ３０９へ進む。そこでmasked_dataから入力された値である２と、data_bufに格納されている値である４を比較し、値が異なっているので図４のステップＳ４０１に進む。
【０１５３】
ステップＳ４０１によりv_outからdata_bufの値である４（04h）を出力し、numから入力されているデータ連続数が１であるためにステップＳ４０２からステップＳ４０８に進む。ここでの内部変数cntの値は1（01h）であるため、ステップＳ４０８からステップＳ４０６に進み、numの値を内部変数cntに加算する（これによりcntは２となる）。そしてステップＳ４０７でmasked_dataから入力された値をdata_bufに格納した後、図３のステップ３０６に戻る。。
【０１５４】
タイミングｔ１５では、データ結合部はタイミングｔ１４と同様の動作を行う。これによりv_outからは２（02h）が出力され、内部変数cntの値は１加算されて３に、data_bufの値はmasked_dataから入力された０（00h）となる。
【０１５５】
タイミングｔ１７においてnumから４（04h）、masked_dataから０（00h）が入力されると、データ分割部はステップＳ３０７からＳ３０８へ進み、内部変数continueが０であるためステップＳ３０９へ進む。そこでmasked_dataから入力された値である０と、data_bufに格納されている値である０を比較し、値が同じであるので図５のステップＳ５０１に進む。
【０１５６】
ステップＳ５０１により内部変数continueの値を「同じデータ連続する」部分を表す１に設定し、ステップＳ５０２により内部変数cntの値を参照する。このときのcntの値は３であるためステップＳ５０３に進み、ｃｎｔ値から「異なるデータが連続する」数を表す長さ情報を算出する。具体的にはcnt値である３から２を減じた値である１（01h）を長さ情報としてv_outから出力する。
【０１５７】
そしてステップＳ５０３によりcntに１を設定し、ステップＳ５０４によりnumから入力されたデータ連続数である４をcnt値に加算して５（05h）として、図３のステップＳ３０６に戻る。
【０１５８】
タイミングｔ１８においてend（118）から１が入力されると、データ結合部ではステップＳ３０６においてデータ入力が終了したと判断して動作を終了するための処理のために図８のステップＳ８０１に進む。
【０１５９】
ステップＳ８０１においてdata_bufに格納されたデータ値をv_outから出力した後、ステップＳ８０２により内部変数continueの値を参照する。タイミングｔ１８におけるcontinueの値は１であるため、ステップＳ８０５に進んでcntの値から「同じデータが連続する」部分を表す長さ情報を生成する。具体的にはcnt値である5（05h）を下式に用いて長さ情報として252(FCh)を算出してv_outから出力する。

２５６－ｃｎｔ＋１
その後、後段にデータの出力が終了した事を知らせるためにステップＳ８０４においてv_endから１を出力して動作を停止する。
【０１６０】
尚、上記説明した動作はそれぞれ、ステップＳ８０１のdata_bufの値の出力がタイミングｔ18、ステップＳ８０５による長さ情報の出力がタイミングｔ19、ステップＳ８０４のv_endからの１の出力はタイミングｔ２０において行われる。
【０１６１】
また、タイミングｔ18でendから１が入力された状態については、データ結合部のv_outからデータ部および長さ情報の二つのデータを出力する必要があるため、タイミングｔ１８においてデータ結合部のreadyから１を出力し、タイミングｔ18で入力されたendの状態をタイミングｔ19まで保持することにより、データ結合部はデータ入力の終了を了承する。
【０１６２】
上記説明した動作により、実施形態におけるデータ結合部１０３は通常のパックビッツ符号化データの並び順序である「連続する数＋データ値」又は「異なるデータが連続する数＋異なるデータ列」とは逆に、「データ値＋連続する数」又は「異なるデータ列＋異なるデータが連続する数」という順序でパックビッツ符号化データを出力する。
【０１６３】
データ出力部１０４は、データ結合部１０３から出力された符号化データを並び換え、通常のパックビッツ符号化データの順序になるように長さ情報とデータ部の並び順を入れ替えて出力する。
【０１６４】
実施形態におけるデータ出力部１０４が行う処理手順として示すと、図９のフローチャートのようになろう。
【０１６５】
データ出力部は動作を開始すると内部変数の初期設定を行う（ステップＳ９０１）。本実施形態におけるデータ出力部には内部変数として、v_outから入力される長さ情報を格納するバッファを示すL_locate、同様にv_outから入力されるデータ部のデータ値を格納するバッファを示すD_locate、及び長さ情報及びデータ部のデータ値を格納するための１６個の８ビットのデータバッファdata_buf[15]～data_buf[0]、data_buf[15]～data_buf[8]の内容を出力する時に出力先のアドレスを示すアドレスバッファaddr_buf[1]及びdata_buf[7]～data_buf[0]の内容を出力する時に出力先のアドレスを示すアドレスバッファaddr_buf[0]が存在する。
【０１６６】
ステップＳ９０１ではこれらの内部変数を、
L_locate ＝ 0
D_locate ＝ 1
addr_buf[0] ＝ 0
addr_buf[1] ＝ 1
data_buf[15]～data_buf[0] ＝ 0
のように設定する。
【０１６７】
内部変数の初期化が終了すると、データ出力部はデータ結合部からのデータの入力が終了しているかどうかの判断を行う（ステップＳ９０２）。データの入力終了の判断は、本実施形態ではv_endから１が入力されているかどうかで判断し、１が入力されているとデータの入力終了とする。
【０１６８】
ステップＳ９０２においてデータ入力が終了していない（v_end = 0）と判断されると、データ出力部はv_outからデータが入力されるのを待つ。本実施形態ではv_outから長さ情

報が入力される場合はl_validから１が、データ部のデータ値が入力される場合はd_validから１が入力されるため、具体的にはデータ出力部はl_validの値を参照（ステップＳ９０３）及びd_validの値を参照（ステップＳ９０４）することによりv_outからのデータ入力を待つ。そしてv_endから１が入力されてデータ入力が終了するか、又はl_valid又はd_validが１になりv_outからデータが入力されるまで、図９のステップＳ９０２乃至９０４を繰り返す。

【０１６９】

ステップＳ９０３によりl_validから１が入力されたことを検知すると、データ出力部はv_outから入力される長さ情報をL_lodateが示す番号のdata_buf（以下data_buf[L_lodate]と示す）に格納し（ステップＳ９０５）、D_lodateが示していた値を新たな長さ情報格納位置としてL_locateに格納し、D_locateはD_locateが格納していた値に１を加える（ステップＳ９０６）。

【０１７０】

また、ステップＳ９０７では、ステップＳ９０６において設定されたL_locateの値を参照する。本実施形態ではL_locteの値より小さい番号のデータバッファには既にデータが格納されているため、L_locateの値が７より大きい場合はdata_buf[7]～data_buf[0]の全てのデータバッファに値が格納されていることになる。

【０１７１】

そのためdata_buf[7]～data_buf[0]の格納先を表すアドレスバッファaddr_buf[0]の値をaddressから出力すると共に、reencode_dataからdata_buf[7]～data_buf[0]の値を出力する（ステップＳ９０８）。

【０１７２】

そしてaddr_buf[1]に格納されているアドレスを新たなアドレスとしてaddr_buf[0]に格納し、addr_buf[1]はaddr_buf[1]に格納していた値に１を加える。また、data_buf[15]～data_buf[8]に格納された値をdata_buf[7]～data_buf[0]に代入し、data_buf[15]～data_buf[8]は０に初期化する。さらにD_locate及びL_locateの値からそれぞれ８を減じる（ステップＳ９０９）。

【０１７３】

これにより次のデータの取得準備が終了したのでデータ出力部はステップＳ９０５に戻りv_endから１が入力されるかv_outから新たなデータが入力されるのを待つ。

【０１７４】

尚、ステップＳ９０７において参照したL_locateの値が７以下である場合は、そのままステップＳ９０２に戻り、v_endから１が入力されるかv_outから新たなデータが入力されるのを待つ。

【０１７５】

また、ステップＳ９０３においてl_validの値が０であり、かつステップＳ９０４においてd_validから１が入力されたことを検知すると、データ出力部はv_outから入力されるデータ部のデータ値をD_lodateが示す番号のdata_buf（以下data_buf[D_lodate]と示す）に格納し（ステップＳ９１０）、D_locateが格納していた値に１を加える（ステップＳ９１１）。

【０１７６】

また、ステップＳ９１１において設定されたD_locateの値を参照する（ステップＳ９１２）。本実施形態ではステップＳ９０２、９０３、９０４におけるL_locteは７以下の値しか取らないため、D_locateの値が１５より大きい場合はdata_buf[15]～data_buf[8]の全てのデータバッファに値が格納されていることになる。

【０１７７】

そのためdata_buf[15]～data_buf[8]の格納先を表すアドレスバッファaddr_buf[1]の値をaddressから出力すると共に、reencode_dataからdata_buf[15]～data_buf[8]の値を出力する（ステップＳ９１３）。

【０１７８】

そしてaddr_buf[1]に格納していた値に１を加え、data_buf[15]～data_buf[8]の内容を０に初期化すると共にD_locateの値から８を減じる（ステップＳ９１４）。

【０１７９】

これにより次のデータの取得準備が終了したのでデータ出力部はステップＳ９０２に戻りv_endから１が入力されるかv_outから新たなデータが入力されるのを待つ。

【０１８０】

尚、ステップＳ９１２において参照したD_locateの値が15以下である場合は、そのままステップＳ９０２に戻り、v_endから１が入力されるかv_outから新たなデータが入力されるのを待つ。

【０１８１】

また、ステップＳ９０２においてv_endから１が入力されてデータ入力が終了すると、データ出力部はL_locateの値を参照する（ステップＳ９１５）。L_locateの値が０以外である場合はデータバッファdata_buf[7]～data_buf[0]のいくつかのバッファの中に未出力のデータが存在しているので、data_buf[7]～data_buf[0]の格納先を表すアドレスバッファaddr_buf[0]の値をaddressから出力すると共に、reencode_dataからdata_buf[7]～data_buf[0]の値を出力する（ステップＳ９１６）。

【０１８２】

その後、後段にデータの出力終了を知らせるためにreencode_endから１を出力して（ステップＳ９１７）動作を終了する。

【０１８３】

またステップＳ９１５においてL_locateの値が0であった場合は、データバッファの中に出力されていないデータが無いため、ステップＳ９１７に進みreencode_endから１を出力してデータの出力終了を知らせた後、動作を終了する。

【０１８４】

尚、上記説明におけるステップＳ９０９及びＳ９１４においてreencode_dtaからデータを、またaddressからアドレスを出力する際にはreencode_validを１にすることを付け加えておく。

【０１８５】

図１２、図１３は、本実施形態におけるデータ出力部１０４に、図１１に示すタイミングでd_valid、l_valid、v_out及びv_endが入力された場合のデータ出力部の動作を表す図であり、内部変数及び出力信号の値の遷移を示しめしている。

【０１８６】

図１２における先頭にあるのは、図１１のタイミングｔ０におけるデータ出力部の状態である。同図同箇所において、1201は図１１における時間（タイミング）、1202は1201で表した時間におけるデータ出力部への入力値を表している。、また、1202は1201で表した時間における内部変数の値を、1203は同じ様に1201で表した時間におけるデータ出力部の出力値を表している。尚、図１２のタイミングｔ０では図９のステップＳ901において内部変数を初期化した状態が示されている。尚、data_buf[0]～data_buf[15]の値は０に設定されるが、タイミングｔ０ではv_outから入力されるデータと区別をするためにdata_bufの値は空欄で示している。

【０１８７】

タイミングｔ７では、v_outからデータ部のデータ値が入力された状態を表している。この時のデータ出力部１０４はステップＳ９０４によりd_validが１であることを検知し、D_locateの値が１であることからステップＳ９１０によりv_outから入力された値をdata_buf[1]に格納すると共に、ステップ９１１によりD_locateの値に１を加算して２とする。その後、ステップＳ９１２によりD_locateの値を参照し、１５以下であるためにステップＳ９０２に戻る。上記動作後は、タイミングｔ８に示す状態になる。

【０１８８】

図１２のタイミングｔ８は、v_outから長さ情報が入力された状態を表している。この時、データ出力部１０４はステップＳ９０３によりｌ_validが１であることを検知し、L_lo

cateの値が０であることからステップＳ９０５によりv_outから入力された値をdata_buf[0]に格納すると共に、ステップＳ９０６によりD_locateに格納されていた値２をL_locateに格納し、D_locateの値２に１を加算して３とする。その後、ステップＳ９０７によりL_locateの値を参照し、７以下であるためにステップＳ９０２に戻る。上記動作後の内部変数の状態はタイミングｔ１２のようになる。

【０１８９】

同様にしてタイミングｔ１２、ｔ１４、ｔ１５、及び、図１３におけるタイミングｔ１８ではd_validから１が入力されるため、D_locateの値が示すdata_bufにv_outから入力されるデータ部のデータ値を格納し、D_locateの値を１加算する。

【０１９０】

また、タイミングｔ１７ではl_validから１が入力されるため、L_locateの値が示すdata_bufにv_outから入力された長さ情報を格納し、D_locateの値をL_locateへ格納してD_locateの値を１加算する。

【０１９１】

タイミングｔ１９ではv_outから長さ情報が入力されている。このときL_locateの値は７であるため、ステップＳ９０５によりv_outから入力された値FChはdata_buf[7]に格納され、その後ステップＳ９０６によりL_locateにはD_locateの値である９が、またD_locateには10が格納される。

【０１９２】

この時、ステップＳ９０７においてL_locateを参照するとL_locateの値は７以上であるため、ステップＳ９０８によりaddressからaddr_buf[0]の値である０が、またreencode_dataからはdata_buf[0]～data_buf[7]の内容を出力する。

【０１９３】

その後、ステップＳ９０９によりaddr_buf[0]の内容をaddr_buf[1]に格納されている１に設定し、addr_buf[1]の値は１を加算して２とする。またdata_buf[8]～data_buf[15]に格納された値をdata_buf[0]～data_buf[7]に設定して、data_buf[8]～data_buf[15]を０に初期化する（図１２、図13内では空欄で表す）。さらにL_locate及びD_locateの値から８を減じてL_locateの値を１に、D_locateの値を２にする。

【０１９４】

尚、上記ステップＳ９０５～Ｓ９０９の動作結果は図１３のタイミングｔ２０に示されている。また、上記説明においてaddress及びreencode_dataからアドレス及びデータを出力する際はreencode_validからも１を出力する。

【０１９５】

図１３のタイミングｔ２０において、v_endから１が入力されている状態を表している。この時データ出力部１０４はステップＳ９０２によりv_endが１であることを検知し、ステップＳ９０５によりL_locateの値を参照する。タイミングｔ２０ではL_locateの値は１であるため、ステップＳ９１６によりdata_buf[0]～data_buf[7]に残ったデータの出力を行う。そのため、reencode_validから１を出力すると共に、addressからはaddr_buf[0]の値である１をreencode_dataからはdata_buf[0]～data_buf[7]の値を出力する。

【０１９６】

その後、ステップＳ９１７によりreencode_endから１を出力してデータ出力部の動作を終了する。

【０１９７】

上記説明において、ステップＳ９１６における動作結果は図１３のタイミングｔ２１に、ステップＳ９１７における動作結果は図１３のタイミングｔ２２のようになる。

【０１９８】

以上説明した様に本実施形態によれば、パックビッツ符号化装置ではパックビッツ符号化データを入力し、パックビッツ符号化データのままデータ部に対して所定の処理を行った後で再度パックビッツ符号化処理（パックビッツ符号の再構成）を行う事が可能となる。本実施形態では具体例として、図１０においてencode_dataから入力された１４個のパッ

クビッツ符号化データが、データ処理部により最下位ビットを０にすることにより図１１のv_outに示す様に９個のパックビッツ符号化データに圧縮する事が出来るようになる。

【０１９９】

なお、実施形態では、像域情報をパックビッツ符号化する場合について説明したが、画像データである各画素値を符号化対象としても構わない。この場合、先に示したオーバーフローの回数とマスクするビットの関係は、画素値のＬＳＢから上位に向かうＮビット（Ｎ＝オーバーフロー回数）をマスクするようにすれば良いであろう。

【０２００】

また、実施形態では、パックビッツ符号化を例にして説明したが、同じデータの連続する数とそのデータ、異なるデータが連続する数とそのデータ列で表現される符号化に対して適用できるものであるから、上記実施形態によって本願発明が限定されるものではない。

【０２０１】

＜第２の実施形態＞

図１６は本発明のパックビッツ再符号化部の他の例（第２の実施形態）である。

【０２０２】

図１６のパックビッツ符号化装置はencode_dataからパックビッツ符号化データだけでなくパックビッツ符号化を行う前の生データの入力も可能にするため、データ分割部１４０１にraw_data信号（1410）が追加されている。

【０２０３】

本第２の実施形態のパックビッツ符号化装置ではraw_dataから０が入力されると、データ分割部１４０１は図２に示す様にencode_data（111）からパックビッツ符号化データが入力された場合の動作を行い、raw_dataから１が入力されるとencode_data（111）からパックビッツ符号化を行う前の生データが入力された場合の動作を行う。

【０２０４】

生データが入力された場合のデータ分割部１４０１の動作を具体的に説明すると、encode_data（111）から生データが入力されると、入力されたデータはdata（115）からデータ処理部１０２へ出力し、それと同時にnum（116）からデータ連続数として１（01h）を出力する。

【０２０５】

これによりデータ結合部１０３及びデータ処理部１０４では、データ分割部１４０１から出力された後、データ処理部１０２で処理されたデータ値を「データ連続数が１」であるデータとしてパックビッツ符号化処理を行う。

【０２０６】

以上の構成により、図１６のパックビッツ符号化装置はパックビッツ符号化データ及び生データのどちらを入力しても、パックビッツ符号化データを生成して出力することが可能になる。

【０２０７】

以上述べたように本第１、第２の実施形態によれば、パックビッツ符号化を行った符号化データを複号処理することなく、パックビッツ符号化データを長さ情報とデータ部に分割し、データ部のデータ値に対して処理を行い、処理後のデータ値と長さ情報を用いて新たなパックビッツ符号化データを再構成することが可能となり、再符号化する際に多くのメモリを必要とせず、高速な再符号化が行えるようになる。

【０２０８】

また、実施形態では、複写機に適用させた例を説明したが、本願発明はこれに限定されるものではない。

【０２０９】

更に、実施形態では、再符号化する際に、マスクする（０にする）ビット位置を決定する例を示したが、１にセットすることで対処しても構わない。すなわち、正論理／負論理のいずれを採用しても構わない。要するに、再符号化する際に、データの取り得る種類の数を減らす方向にビット変更すれ良い。なぜなら、この結果、同じデータの連続する割合が

高くできることになり、より高い圧縮率が期待できるからである。
【０２１０】
また、実施形態で説明したパックビッツ再符号化部１５が有する夫々の構成要素をフローチャートを用いて説明した。かかる説明から明らかなように、その機能と同機能をコンピュータプログラムで実現させても構わない。すなわち、本発明はコンピュータプログラムもその範疇とする。また、通常、コンピュータプログラムは、それが記憶されたＣＤＲＯＭ等のコンピュータ可読記憶媒体を装置にセットし、システムにコピーもしくはインストールすることで実行可能となるわけであるから、本発明はかかるコンピュータ可読記憶媒体をもその範疇とするのは明らかである。
【０２１１】
以上であるが、上記実施形態での記載に従って本発明にかかる実施態様を列挙すれば次の通りである。
【０２１２】
［実施態様１］　同じデータの連続数を示す連続数コード部と前記データを示すデータ部、及び、異なるデータ列の連続数を示す連続数コード部と前記異なるデータ列を示すデータ部のデータ形式で表現される符号化データを、前記データ形式に再符号化する符号化方法であって、
前記データ形式の符号化データのストリームを入力する工程と、
入力したストリームから、連続数コードとデータ部とに分割する工程と、
分割したデータ部の所望とするビットを変更する工程と、
変更したデータ部と前記連続数コード部に基づき、前記データ形式に再構成する工程と、
再構成したデータを再符号化データストリームとして出力する工程と
を備えることを特徴とする符号化方法。
【０２１３】
［実施態様２］　更に、前記変更する工程に対し、変更するビット位置を指定する工程を有することを特徴とする実施態様１に記載の符号化方法。
【０２１４】
［実施態様３］　前記指定工程は、データ部の所定のビットの情報に基づいて変更するビット位置を指定することを特徴とする実施態様２に記載の符号化方法。
【０２１５】
［実施態様４］　前記符号化データのデータ部は、画像データの画素の複数の像域属性のフラグビットで構成されることを特徴とする実施態様１乃至３のいずれか１項に記載の符号化方法。
【０２１６】
［実施態様５］　前記符号化データは、パックビッツ符号化データであることを特徴とする実施態様１乃至４のいずれか１項に記載の符号化方法。
【０２１７】
［実施態様６］　同じデータの連続数を示す連続数コード部と前記データを示すデータ部、及び、異なるデータ列の連続数を示す連続数コード部と前記異なるデータ列を示すデータ部のデータ形式で表現される符号化データを、前記データ形式に再符号化する符号化装置であって、
前記データ形式の符号化データのストリームを入力する入力手段と、
入力したストリームから、連続数コードとデータ部とに分割する分割手段と、
分割したデータ部の所望とするビットを変更する変更手段と、
変更したデータ部と前記連続数コード部に基づき、前記データ形式に再構成する再構成手段と、
再構成したデータを再符号化データストリームとして出力する出力手段と
を備えることを特徴とする符号化装置。
【０２１８】
［実施態様７］　同じデータの連続数を示す連続数コード部と前記データを示すデータ部

、及び、異なるデータ列の連続数を示す連続数コード部と前記異なるデータ列を示すデータ部のデータ形式で表現される符号化データを、前記データ形式に再符号化する符号化装置として機能するコンピュータプログラムであって、
前記データ形式の符号化データのストリームを入力する入力手段と、
入力したストリームから、連続数コードとデータ部とに分割する分割手段と、
分割したデータ部の所望とするビットを変更する変更手段と、
変更したデータ部と前記連続数コード部に基づき、前記データ形式に再構成する再構成手段と、
再構成したデータを再符号化データストリームとして出力する出力手段と
して機能することを特徴とするコンピュータプログラム。

【０２１９】

［実施態様８］　実施態様７に記載のコンピュータプログラムを格納することを特徴とするコンピュータ可読記憶媒体。

【０２２０】

なお、上記実施態様６、７、８に対して、実施態様２乃至５が付加可能であることは明らかである。

【０２２１】

【発明の効果】

以上説明したように本発明によれば、パックビッツ符号化データの如く、同じデータの連続数を示す連続数コード部と前記データを示すデータ部、及び、異なるデータ列の連続数を示す連続数コード部と前記異なるデータ列を示すデータ部のデータ形式で表現される符号化データを、復号処理することなく、同じデータ形式に再符号化し、圧縮率を向上させることが可能になる。

【図面の簡単な説明】

【図１】実施形態におけるパックビッツ再符号化部のブロック構成図である。

【図２】実施形態におけるデータ分割部１０１の動作処理内容を示すフローチャートである。

【図３】実施形態におけるデータ結合部１０３の動作処理内容を示すフローチャートである。

【図４】実施形態におけるデータ結合部１０３の動作処理内容を示すフローチャートである。

【図５】実施形態におけるデータ結合部１０３の動作処理内容を示すフローチャートである。

【図６】実施形態におけるデータ結合部１０３の動作処理内容を示すフローチャートである。

【図７】実施形態におけるデータ結合部１０３の動作処理内容を示すフローチャートである。

【図８】実施形態におけるデータ結合部１０３の動作処理内容を示すフローチャートである。

【図９】実施形態におけるデータ出力部１０４の動作処理内容を示すフローチャートである。

【図１０】実施形態におけるデータ分割部１０１及びデータ処理部１０２の入出力タイミングチャートである。

【図１１】実施形態におけるデータ結合部１０３の入出力タイミングチャートである。

【図１２】実施形態におけるデータ出力部１０４の動作にともなう内部状態の遷移図である。

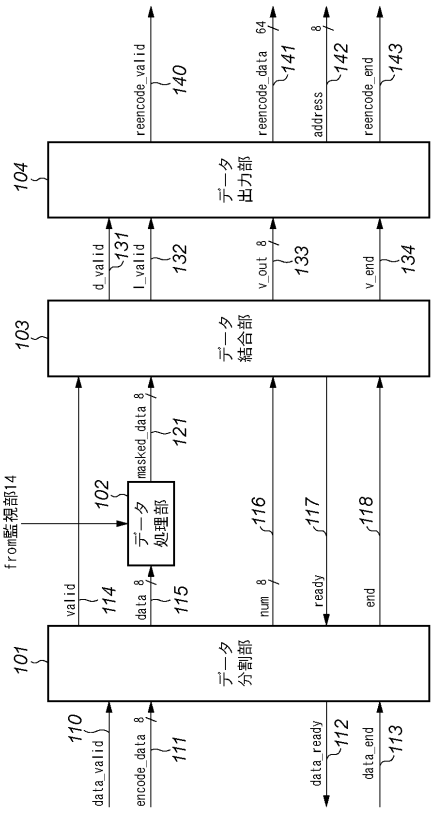
【図１３】実施形態におけるデータ出力部１０４の動作にともなう内部状態の遷移図である。

【図１４】実施形態が適用するデジタル複写機のブロック構成図である。

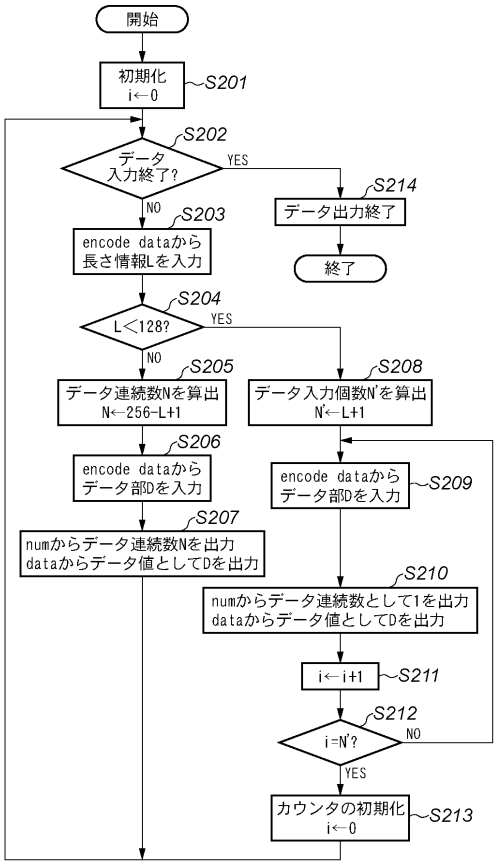
【図１５】図１４における像域情報符号化部のブロック構成図である。

【図１６】第２の実施形態におけるパックビッツ再符号化装置のブロック構成図である。

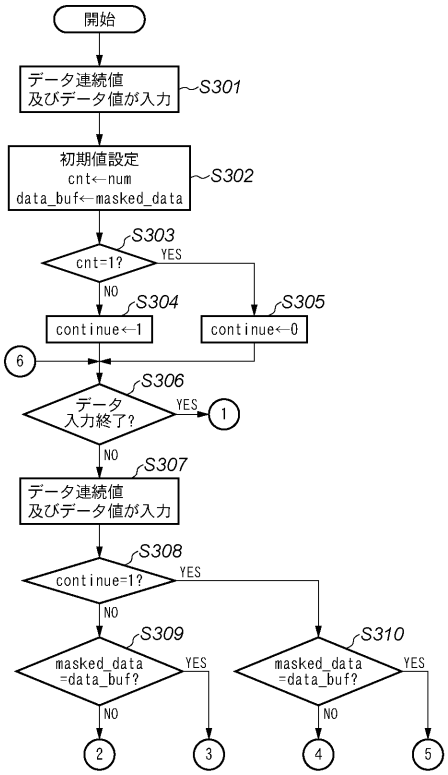
【図１】



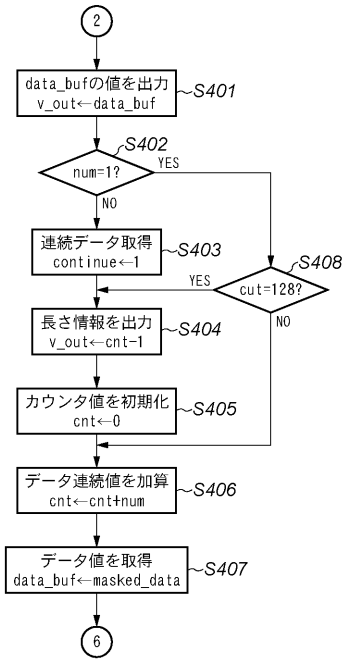
【図２】



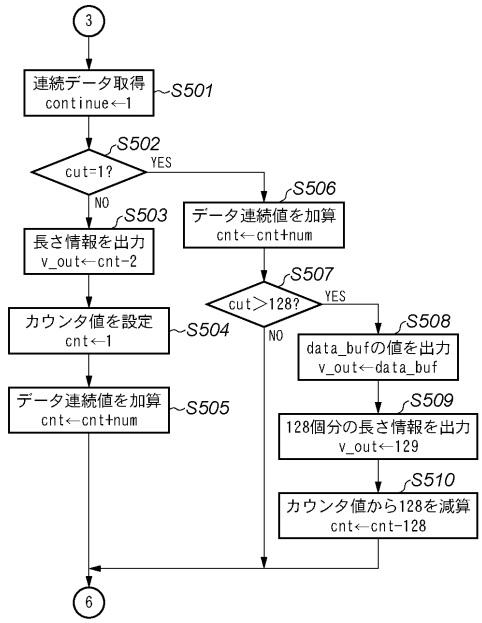
【図３】


開始
データ連続値
及びデータ値が入力
S301
初期値設定
cnt←num
data_buf←masked_data
S302
cnt=1?
S303
YES
NO
continue←1
S304
continue←0
S305
6
データ
入力終了?
S306
YES
1
NO
データ連続値
及びデータ値が入力
S307
continue=1?
S308
YES
NO
masked_data
=data_buf?
S309
masked_data
=data_buf?
S310
2
3
4
5

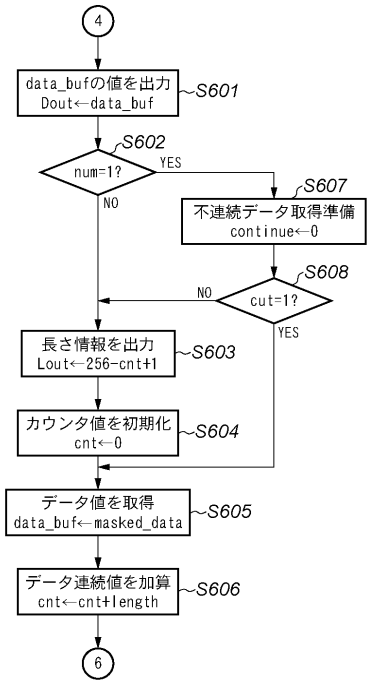
【図４】


2
data_bufの値を出力
v_out←data_buf
S401
num=1?
S402
YES
NO
連続データ取得
continue←1
S403
cut=128?
S408
YES
NO
長さ情報を出力
v_out←cnt-1
S404
カウンタ値を初期化
cnt←0
S405
データ連続値を加算
cnt←cnt+num
S406
データ値を取得
data_buf←masked_data
S407
6

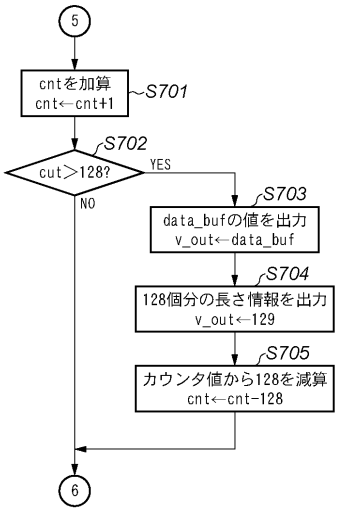
【図５】


3
連続データ取得
continue←1
S501
cut=1?
S502
YES
NO
長さ情報を出力
v_out←cnt-2
S503
データ連続値を加算
cnt←cnt+num
S506
cut＞128?
S507
YES
NO
カウンタ値を設定
cnt←1
S504
data_bufの値を出力
v_out←data_buf
S508
データ連続値を加算
cnt←cnt+num
S505
128個分の長さ情報を出力
v_out←129
S509
カウンタ値から128を減算
cnt←cnt-128
S510
6

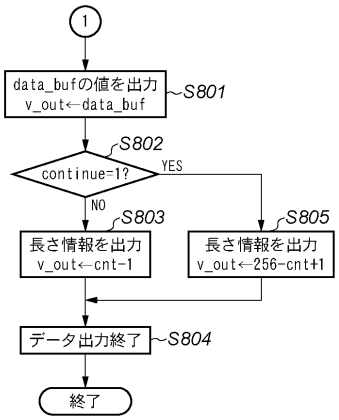
【図６】


4
data_bufの値を出力
Dout←data_buf
S601
num=1?
S602
YES
NO
不連続データ取得準備
continue←0
S607
cut=1?
S608
NO
YES
長さ情報を出力
Lout←256-cnt+1
S603
カウンタ値を初期化
cnt←0
S604
データ値を取得
data_buf←masked_data
S605
データ連続値を加算
cnt←cnt+length
S606
6

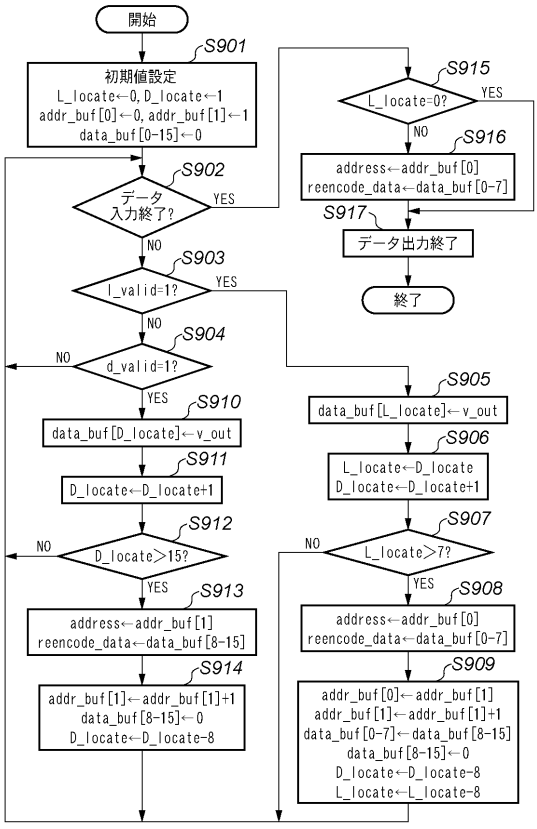
【図７】


5
cntを加算
cnt←cnt+1
S701
S702
cut＞128?
YES
NO
S703
data_bufの値を出力
v_out←data_buf
S704
128個分の長さ情報を出力
v_out←129
S705
カウンタ値から128を減算
cnt←cnt-128
6

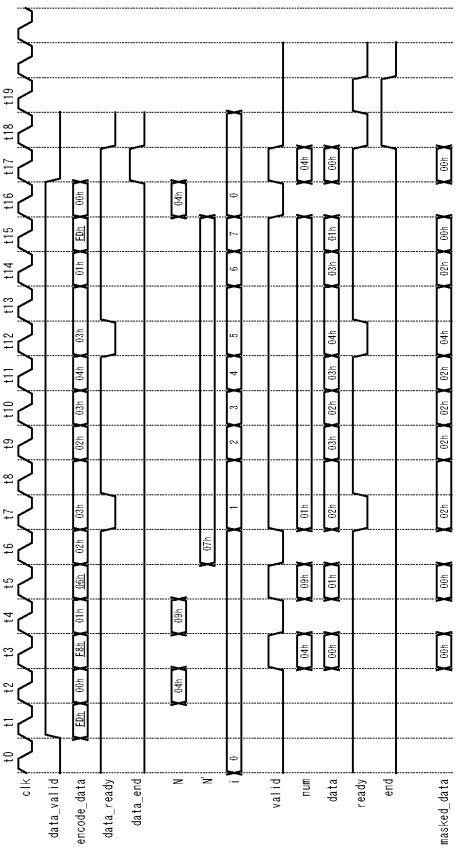
【図８】


1
data_bufの値を出力
v_out←data_buf
S801
S802
continue=1?
YES
NO
S803
長さ情報を出力
v_out←cnt-1
S805
長さ情報を出力
v_out←256-cnt+1
データ出力終了
S804
終了

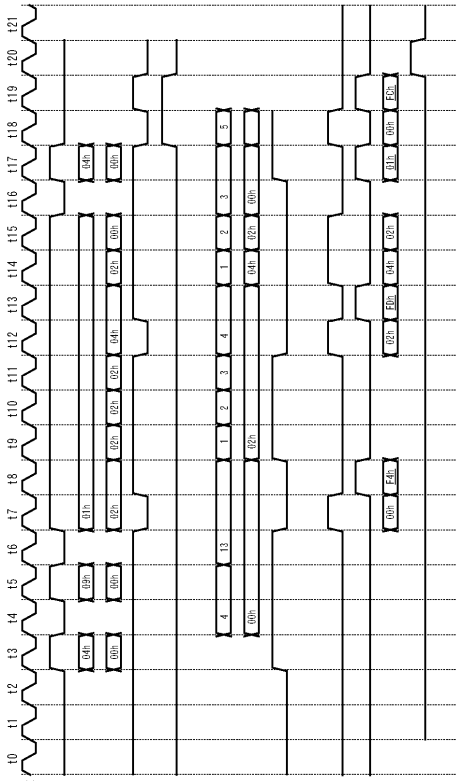
【図９】


開始
S901
初期値設定
L_locate←0, D_locate←1
addr_buf[0]←0, addr_buf[1]←1
data_buf[0-15]←0
S902
データ
入力終了?
YES
NO
S903
l_valid=1?
YES
NO
S904
d_valid=1?
NO
YES
S910
data_buf[D_locate]←v_out
S911
D_locate←D_locate+1
S912
D_locate＞15?
NO
YES
S913
address←addr_buf[1]
reencode_data←data_buf[8-15]
S914
addr_buf[1]←addr_buf[1]+1
data_buf[8-15]←0
D_locate←D_locate-8
S915
L_locate=0?
YES
NO
S916
address←addr_buf[0]
reencode_data←data_buf[0-7]
S917
データ出力終了
終了
S905
data_buf[L_locate]←v_out
S906
L_locate←D_locate
D_locate←D_locate+1
S907
L_locate＞7?
NO
YES
S908
address←addr_buf[0]
reencode_data←data_buf[0-7]
S909
addr_buf[0]←addr_buf[1]
addr_buf[1]←addr_buf[1]+1
data_buf[0-7]←data_buf[8-15]
data_buf[8-15]←0
D_locate←D_locate-8
L_locate←L_locate-8

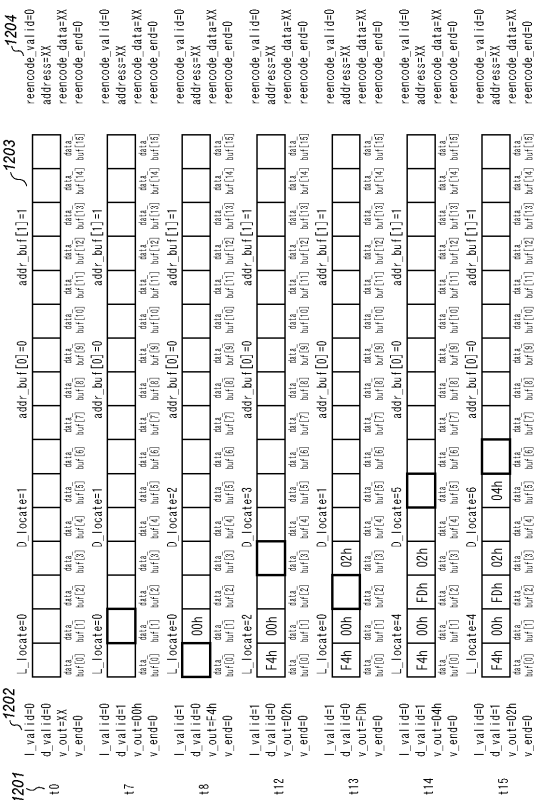
【図１０】


t0
t1
t2
t3
t4
t5
t6
t7
t8
t9
t10
t11
t12
t13
t14
t15
t16
t17
t18
t19
clk
data_valid
encode_data
data_ready
data_end
N
N'
i
valid
num
data
ready
end
masked_data

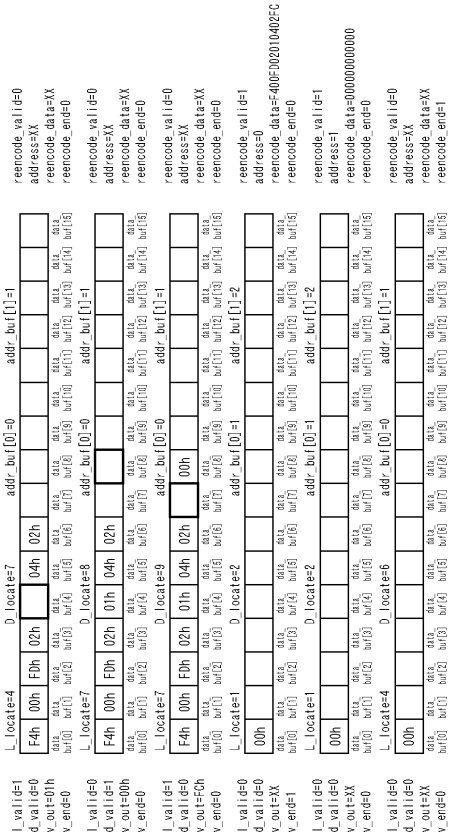
【図１１】



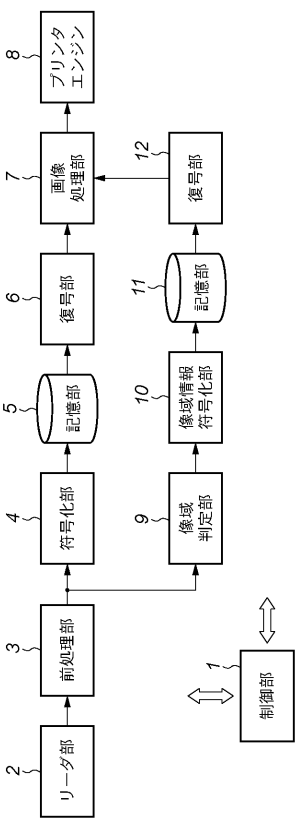
【図１２】



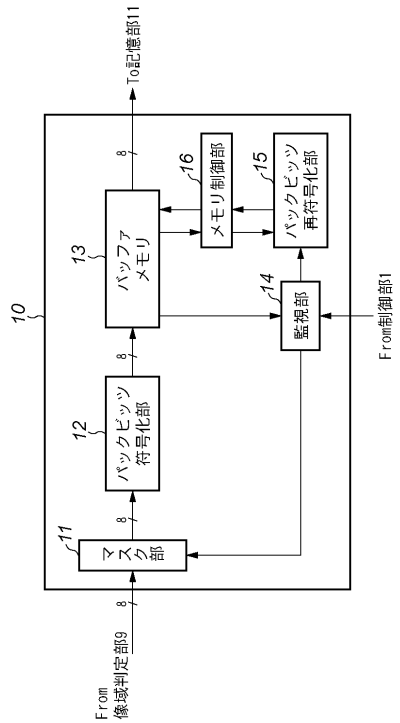
【図１３】



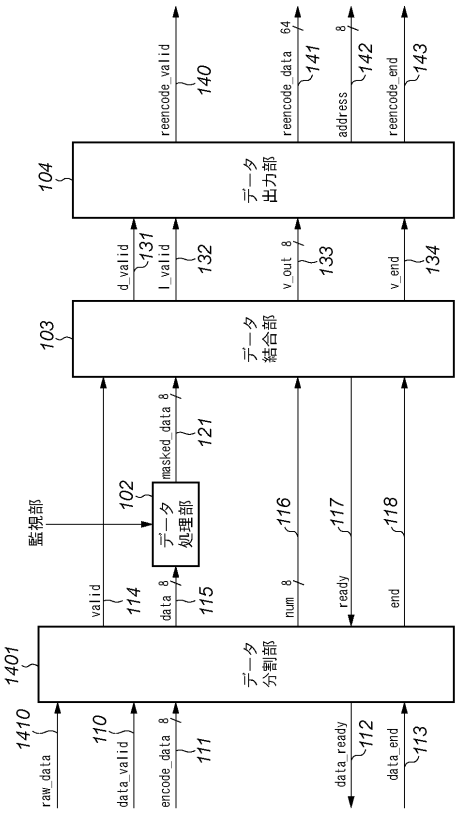
【図１４】



【図１５】


10
11
マスク部
12
バックビッツ符号化部
13
バッファメモリ
14
監視部
15
バックビッツ再符号化部
16
メモリ制御部
8
From像域判定部9
To記憶部11
From制御部1

【図１６】


1401
データ分割部
103
データ結合部
104
データ出力部
102
データ処理部
監視部
raw_data
1410
data_valid
110
encode_data 8
111
valid
114
data 8
115
masked_data 8
121
d_valid
131
l_valid
132
num 8
116
ready
117
end
118
v_out 8
133
v_end
134
data_ready
112
data_end
113
reencode_valid
140
reencode_data 64
141
address 8
142
reencode_end
143

フロントページの続き

審査官　北村　智彦

(56)参考文献　特開２００３－０６９８３５（ＪＰ，Ａ）
特開平１１－２８９４６０（ＪＰ，Ａ）
特開平０８－０９６１４２（ＪＰ，Ａ）
特開平１１－０３１９７５（ＪＰ，Ａ）
特開２００１－００８０４２（ＪＰ，Ａ）
特開２０００－２４２４５０（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
H03M3/00-11/00
H04N 1/419
H04N 7/26