(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第5075136号
(P5075136)

(45)発行日　平成24年11月14日(2012.11.14)　(24)登録日　平成24年8月31日(2012.8.31)

(51)Int.Cl.			F I		
H03M	7/40	(2006.01)	H03M	7/40	
H04N	7/26	(2006.01)	H04N	7/13	Z

請求項の数 12　(全 33 頁)

(21)出願番号　特願2009-8242(P2009-8242)
(22)出願日　平成21年1月16日(2009.1.16)
(65)公開番号　特開2010-166445(P2010-166445A)
(43)公開日　平成22年7月29日(2010.7.29)
審査請求日　平成24年1月10日(2012.1.10)

(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
(74)代理人　100130409
弁理士　下山　治
(74)代理人　100134175
弁理士　永川　行光

最終頁に続く

(54)【発明の名称】復号装置及びその制御方法

(57)【特許請求の範囲】
【請求項１】
複数画素で構成される画素ブロック単位に可変長符号化することで生成された符号化データを復号する復号装置であって、
符号化データから符号語の先頭ビットを頭だしするためのシフタと、
１つのアドレスに１ないし複数の符号語のデコード値を格納する第１のテーブルと、
前記シフタのシフト量を格納する為の第２のテーブルと、
前記１ないし複数の符号語のデコードデータ長を生成する為の第３のテーブルと、
前記符号化データから前記第１のテーブルのアドレスを生成する為の第１のデコーダと、
前記符号化データから前記第２及び第３のテーブルのアドレスを生成する為の第２のデコーダと、
前記複数の符号語のデコード値を一定の固定ビット数分のデータに結合又は分割して出力する為の出力部と
を有することを特徴とする復号装置。
【請求項２】
前記可変長符号化はランレングス符号化であることを特徴とする請求項１に記載の復号装置。
【請求項３】
前記第１のテーブルは、前記１ないし複数の符号語のデコードデータのうち最後のデコ

ードデータを格納ビット数まで連続させるか、最後のデコードデータを格納ビット数で打ち切った値を格納することを特徴とする請求項２に記載の復号装置。
【請求項４】
前記第１のテーブルの格納データは、直前に位置する画素のデコード値が１又は０のいずれか一方に仮定した値を格納することを特徴とする請求項２又は３に記載の復号装置。
【請求項５】
前記第１のテーブルに基づくデコード値は、直前に位置する画素のデコード値が前記仮定した値と等しい場合はそのまま出力し、前記仮定した値と異なる場合は前記第１のテーブルの値を反転して出力することを特徴とする請求項４に記載の復号装置。
【請求項６】
前記出力部は、前記第１のテーブルに格納されているデータが前記第３のテーブルより生成されるデータ長より少ない場合は、前記第１のテーブルより生成されたデコードデータの最終ビットを前記データ長になるまで延長させて出力することを特徴とする請求項１に記載の復号装置。
【請求項７】
前記第２のデコーダは、前記１ないし複数の符号語の所定のビットよりデコードして、アドレスを生成することを特徴とする請求項１に記載の復号装置。
【請求項８】
前記第３のテーブルの格納データは、前記デコードデータのデータ長を算出する為のベース値を格納することを特徴とする請求項１に記載の復号装置。
【請求項９】
前記１ないし複数の符号語のデコード後のデータ長は、前記第３のテーブルからの前記ベース値と前記１ないし複数の符号語の前記第２のデコーダの値に応じたビット位置の値の総和で算出されることを特徴とする請求項８に記載の復号装置。
【請求項１０】
複数画素で構成される画素ブロック単位に可変長符号化することで生成された符号化データを復号するため、
符号化データから符号語の先頭ビットを頭だしするためのシフタと、
１つのアドレスに１ないし複数の符号語のデコード値を格納する第１のテーブルと、
前記シフタのシフト量を格納するための第２のテーブルと
前記１ないし複数の符号語のデコードデータ長を生成する為の第３のテーブルとを有する復号装置の制御方法であって、
前記符号化データから前記第１のテーブルのアドレスを生成する為の第１のデコード工程と、
前記符号化データから前記第２及び第３のテーブルのアドレスを生成する為の第２のデコード工程と、
前記複数の符号語のデコード値を一定の固定ビット数分のデータに結合又は分割して出力する為の出力工程と
を有することを特徴とする復号装置の制御方法。
【請求項１１】
コンピュータに読み込ませ実行させることで、前記コンピュータを、請求項１乃至９のいずれか１項に記載の復号装置として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１２】
請求項１１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】
【技術分野】
【０００１】
本発明は、符号化データを復号する技術に関するものである。

【背景技術】
【０００２】
最大符号長のビット数個分のテーブルとバレルシフタを用意して、最長符号語のビット数個分出力より１つの復号シンボルを選択することにより、ハフマンデコーダからバレルシフタへのフィードバックをなくして高速動作可能な可変長復号化器が提案されている（特許文献１）。
【特許文献１】特開平９－２８４１４２号公報
【発明の開示】
【発明が解決しようとする課題】
【０００３】
しかしながら従来技術では、テーブルとバレルシフタの数が最大符号長のビット数個分と非常に多く回路規模の増大を招いていた。また、スループットを高くする為には複数シンボルを同時デコードする必要があるが、複数シンボルを同時デコードしようとすると上記最大符号長のビット数が同時デコード数倍に増大し、また、複数シンボルを同時デコードするためのテーブルはシンボル数の同時デコード数乗となるため、回路規模の大幅な増大を招いていた。
【０００４】
本発明はかかる課題に鑑みなされたものであり、回路規模の更なる簡略化と、複数シンボル同時デコードの両立可能な技術を提供しようとするものである。
【課題を解決するための手段】
【０００５】
かかる課題を解決するため、例えば本発明の復号装置は以下の構成を備える。すなわち、
複数画素で構成される画素ブロック単位に可変長符号化することで生成された符号化データを復号する復号装置であって、
符号化データから符号語の先頭ビットを頭だしするためのシフタと、
１つのアドレスに１ないし複数の符号語のデコード値を格納する第１のテーブルと、
前記シフタのシフト量を格納する為の第２のテーブルと、
前記１ないし複数の符号語のデコードデータ長を生成する為の第３のテーブルと、
前記符号化データから前記第１のテーブルのアドレスを生成する為の第１のデコーダと、
前記符号化データから前記第２及び第３のテーブルのアドレスを生成する為の第２のデコーダと、
前記複数の符号語のデコード値を一定の固定ビット数分のデータに結合又は分割して出力する為の出力部とを有する。
【発明の効果】
【０００６】
本発明によれば、回路規模の更なる簡略化と、複数シンボル同時デコードの両立することが可能になる。
【発明を実施するための最良の形態】
【０００７】
以下、添付図面に従って本発明に係る実施形態を詳細に説明する。
【０００８】
実施形態の画像復号装置が復号する対象は、２値画像（１画素１ビット）の符号化データである。そして、複数画素で構成される画素ブロック（実施形態では８×８画素）を単位とする符号化データを順次復号するものである。
【０００９】
そこで先ず、実施形態における画像復号装置の説明に先立ち、画像符号化装置による符号化データ生成処理について説明する。
【００１０】

画像符号化装置は、２値画像データを８×８画素のブロック単位にランレングス符号化する。符号化対象は、イメージスキャナから読取った２値画像データとするが、ネットワーク上に存在する無圧縮の２値画像データファイルや、コンピュータにより処理された２値画像データをその対象としても良く、その入力源の種類は問わない。２値画像の場合、８×８画素のブロック内をラスタースキャンすると、画素値“１”のランと、画素値“０”のランが必ず交互に出現することになる。従って、ブロックの先頭の画素値の値が決まれば、最初のランレングス符号語はその画素値のランを示し、それに後続するランの符号語は、直前のランで示される画素の値に“１”を排他論理和した画素値を示す符号語となる。

【００１１】

図１は、ランレングス符号語とラン長との対応を示している。すなわち、画像符号化装置は、ラン長が０の場合（同じ値を持つ画素が１つのみの場合）は符号語として２ビットの“００”を出力する。また、ラン長が“１”の場合（同じ画素が２つ連続する場合）は、符号語として２ビットの“０１”を出力する。また、ラン長が“２”（同じ値の画素が３個連続する場合）や“３”（同じ値の画素が４個連続すする場合）の場合には、図示の如く３ビットの符号語を出力する。そして、ラン長が４乃至６３である場合（同じ画素が５乃至６４個連続する場合）には、図示の如く、固定の８ビットの符号語“１１００００１００”乃至“１１１１１１１１”を出力する。なお、実施形態では、８×８画素のブロックを単位に符号化するものであるので、ラン長が６３を超えることはない。

【００１２】

また、画像符号化装置は、着目ブロックを符号化している最中に、或る画素から最後の画素まで同じであり、且つ、そのラン長が“４”以上である場合、そのランの符号語を生成せず、符号化データの最後を示す符号語（ＥＯＢ；End Of Block）を示す６ビットの符号語“１１００００”を出力する。ＥＯＢを示す符号語の直前のランの画素値が仮に“１”であった場合、その直後の画素から最後の画素までは“０”の値を持つ画素が連続することが約束される。また、このＥＯＢの符号語“１１００００”は、ラン長が４乃至６３のいずれの符号語にも一致しないので、復号装置ではＥＯＢを検出できる。

【００１３】

また、１ブロックの２値画像データの符号化して得られた符号化データ量が６４ビット以上になってしまうことが起こり得る。本来、２値画像中の８×８画素は６４ビットで構成されるので、これでは圧縮されているとは言えない。そこで、画像符号化装置は、１ブロックの符号化データが６４ビット以上になった場合、その符号化データを破棄し、先頭に非符号化を示す１ビットの識別データ“０”を出力し、それに後続して６４画素（６４ビット）のデータをそのまま出力する。図１５（ａ）は、この場合の符号化データのデータ構造を示している。

【００１４】

一方、１ブロックの符号化データ量が６４ビット未満であった場合、画像符号化装置は、符号化されていることを示す１ビットの識別データ“１”を先ず出力し、その次に先頭画素の値を決定するための１ビットを出力する。そして、その後に符号化データを出力する。図１５（ｂ）はこの場合の符号化データのデータ構造を示している。

【００１５】

以上が実施形態における復号対象の符号化データの生成処理の概要である。次に、実施形態における画像復号装置のブロック構成図を図１４に示す。図示の如く、この装置は、符号化データ入力部１０１、判定部１０２、セレクタ１０３、１０５、及び、復号部１０４で構成される。

【００１６】

このうち、復号部１０４は、符号データから符号語の先頭ビットを頭だしするためのシフタ（３０１）と、１つのアドレスに複数のシンボルデータのデコード値を格納するの第１のテーブルとして機能するテーブル３０３と、シフタのシフト量を格納する為の第２のテーブルとして機能するテーブル３０８と、複数のシンボルデータのデコード値のデータ

長を生成する為の第３のテーブルとして機能するテーブル３０７と、前記符号データから前記第１のテーブルのアドレスを生成する為の第１のデコーダ（もしくは第１のデコード工程）として機能するデコーダ３０２と、前記符号データから前記第２及び第３のテーブルのアドレスを生成する為の第２のデコーダ（もしくは第２のデコード工程）として機能するデコーダ３０６と、前記複数のシンボルデータのデコード値を一定の固定ビット数分のデータに結合又は分割して出力する為の出力部として機能するパッカ３０４とを有する。以下、各構成要素について更に詳しく説明する。

【００１７】

符号化データ入力部１０１は、内部にバッファメモリを有し、符号化データのビットストリームを順次入力しては、そのビットストリームをセレクタ１０３に出力する。

【００１８】

判定部１０２は、１ブロック分の符号化データの符号化データの先頭の識別ビットを判定するものであり、初期段階ではアクティブになっている。そして、判定部１０２は、符号化データの先頭の１ビットが、“０”であるのか“１”であるのかを判定する。すなわち、判定部１０２は、着目ブロックの符号化データが、図１５（ａ）、（ｂ）のいずれであるのかを判定し、その判定結果を制御信号（１ビットで良い）をセレクタ１０３、１０５に出力する。

【００１９】

この結果、着目ブロックが図１５（ａ）の符号化データであると判定した場合、セレクタ１０３は、先頭の識別ビットを除く６４ビットをそのままセレクタ１０５に出力する。そして、セレクタ１０５は、入力した６４ビットのデータを復号結果として出力する。従って、着目ブロックの識別ビットが“０”である場合、セレクタ１０５からは１クロックサイクルで６４ビットのデータを復号結果として出力することができる。

【００２０】

なお、セレクタ１０５は、１ブロックを構成する２値画素の個数をカウントし、１ブロックを構成する６４画素の復号が完了する度に、判定部１０２に対してアクティブにする制御信号を出力する。

【００２１】

一方、判定部１０２が、着目ブロックの符号化データの先頭の識別ビットが“１”である、すなわち、着目ブロックが図１５（ａ）の符号化データであると判定した場合、先頭の識別ビットを除く符号化データを、復号部１０４に供給させるための制御信号をセレクタ１０３に出力する。また、判定部１０２は、セレクタ１０５に対しても、復号部１０４からのデータを選択するように制御信号を供給する。セレクタ１０５は復号部１０４から、出力された２値画素のデータ（詳細は後述するように１６ビット単位となる）をカウントし、６４画素の出力が終えたと判定した場合、判定部１０２をアクティブにする制御信号を出力する。従って、復号部１０４にて６４画素の復号処理中は、判定部１０２はインアクティブな状態であり、セレクタ１０３は符号化データ入力部１０１からの符号化データを復号部１０４に出力しつづける。

【００２２】

上記の通り、図１５（ａ）に示す構造の符号化データについては、１クロックサイクルにて復号できることは容易に理解できよう。そこで、以下では、図１５（ｂ）のデータ構造の符号化データを復号する復号部１０４について更に詳細に説明する。

【００２３】

本実施形態の画像復号装置の復号部１０４は、１ブロックの符号化データを、最大でも１６クロックサイクルで復号するものである。これは、本実施形態の画像符号化装置を利用（内蔵）する複合装置やプリンタ等を設計する設計者等に向けて、本装置は最低でも１クロックサイクル当たり４画素を復号することを保証するものとも言い換えることができる。

【００２４】

以下、本実施形態の第１及び後述する第２の実施形態の復号装置が速度保証を行なうス

ループットの指標に関して説明する。

【００２５】

ここで実施形態における復号部１０４が復号しようとしている符号化データにおいて、ブロックの先頭の１画素の値（１ビット）を除く符号化データのデータ量は６４ビット未満であることが約束されている点に注意されたい（図１５（ｂ）参照）。

【００２６】

処理単位（ここではブロック）の画素数をＮ個（実施形態ではＮ＝６４）とする。そして、１ブロック分の符号化データ量をＰビットとする。目標スループットをＴ画素/クロックサイクル（実施形態では４画素／クロックサイクル）とする。この場合、Ｐ／Ｎは１画素当りの平均符号量である。従って、上記目標スループットＴを満足するには、１クロックサイクル当り、Ｐビット中、（Ｐ／Ｎ）×Ｔビットの符号語について復号処理すればよい。

【００２７】

一方、目標スループットがＴ画素/クロックサイクルであるので、１クロックサイクル当りＴ画素以上処理すれば、上記目標スループットを達成できる。よって、１クロックサイクルあたりの処理符号量をｘビット/クロックサイクル、１クロックサイクルあたりの処理画素数をｙ画素/クロックサイクルとしたとき、目標スループットＴ画素/クロックサイクルを満足する為には、以下に示す式（１）又は式（２）のいずれかを満たせばよいことがわかる。

ｘ ≧ （Ｐ／Ｎ）×Ｔ …（１）

または、

ｙ ≧ Ｔ …（２）

【００２８】

［第１の実施形態］

復号部１０４が復号する１ブロックの符号量はＰ（６４ビット未満）である。従って、１クロックサイクル当たり４画素をデコードするためには、上記（１）から、

ｘ≧（Ｐ／６４）×４＞６４／６４×４＝４

となり、１クロックサイクル当たり４ビット以上を復号処理すればよいことになる。一方、図１に示すように、４ビット以下の符号語（２ビット、３ビットの符号語）が含まれており、これらを１クロックサイクルで復号してしまうと、上記の条件に合致しない。そこで、４ビット未満の符号語については、その次の符号語と連結することで、見かけ上４ビットを超える符号語と見なし、復号処理を行なうものとした。換言すれば、単体で４ビット以上の符号語同士は連結しない。以降、この２つの符号語を結合した符号語を、連結符号語と呼ぶことにする。また、もともとの符号語や、連結符号語を構成する個々の符号語を、連結符号語と区別するため、以降、単体符号語と称することとする。

【００２９】

実施形態の復号処理を理解しやすくため、図２のデコードテーブルについて説明する。同図において、“連結符号語”は先に説明したように複数の符号語を連結したものである。“連結シンボル”は、直前にデコードされた画素の値が“０”であった場合の複数シンボルのデコード値である（以下、連結シンボルデータと称する）。また、説明のためにそれぞれの連結符号語に対して符号語番号を付与している。

【００３０】

図示の如く、符号語番号“０”の連結符号語は、単体符号語“００”が２つ連続する場合と、それをデコードした場合の連結シンボルデータ“１０”を示している。符号語“００”はラン長が０（同じ値を持つ画素が１つのみの場合）を示している。図２は、直前の符号化データが“０”である例を想定しているので、単体符号語“００”が２つ連続する場合の復号結果は、“１０”となる。つまり、連結符号語“００００”を復号すると、その復号結果は２画素であり、その値は“１”、“０”の順になる。

【００３１】

また、符号語番号“１”の連結符号語は単体符号語“００”と単体符号語“０１”が続

けてデコードされる場合であり、デコードされた連結シンボルデータは“１００”となる。つまり、連結符号語“０００１”を復号した場合、その符号化結果は３画素であり、値“１”の画素が１つ、その後に値“０”の画素が２つが生成される。

【００３２】

また、符号語番号“２”の連結符号語は、単体符号語“００”と単体符号語“１００”が続けてデコードされる場合であり、デコードされた連結シンボルデータは“１０００”となる（４画素復号）。また、符号語番号“３”の連結符号語は単体符号語“００”と単体符号語“１０１”が続けてデコードされる場合であり、デコードされた連結シンボルデータは“１００００”となる（５画素復号）。

【００３３】

さらに、符号語番号“４”連結符号語は単体符号語“００”と単体符号語“１１＊＊＊＊＊＊”(ここで“＊＊＊＊＊＊”は“０００１００”（十進数で“４”）乃至“１１１１１０”（十進数で“６２”））が続けてデコードされる場合であり、デコードされた連結シンボルデータは１個の“１”の後、“０”が５乃至６３個だけ続く連結シンボルデータとなる。

【００３４】

同様な方法で、図２には連結シンボルデータを構成する２つの連続する単体符号語に対して単体符号語“００”、“０１”、“１００”、“１０１”、“１１＊＊＊＊＊＊”がそれぞれ出現した場合の組み合わせが列挙されている。但し、符号語番号“２０”においては、“連結符号語”は単体符号語“１１＊＊＊＊＊＊”のみで構成されている。

【００３５】

尚、図示していない例外処理として、単体符号語ＥＯＢ（図１で示す単体符号語“１１０００”）が連結符号語の途中で出現した場合は、１ブロックの残りの画素全てに関して、直前にデコードされたシンボルの値が“０”であった場合は“１”として連結シンボルデータを出力する。また、直前にデコードされたシンボルの値が“１”であった場合は“０”として連結シンボルデータを出力する。

【００３６】

さらに、単体符号語ＥＯＢが連結符号語の先頭で出現した場合(連結符号語自身がＥＯＢの場合)は、直前の画素の値を“１”で排他論理和した結果を、最後の画素まで出力する。図２のテーブルでは、直前の画素データを０と仮定しているため、全て１を出力するようことになる。

【００３７】

また、図２のデコードテーブルにおいては連結符号語の符号長が全て４ビット以上となるように設定してある。従って、１クロックサイクルサイクル当りの処理符号量ｘは４ビット以上となる。復号部１０４が復号する着目ブロックの符号量Ｐは６４ビット未満であることが約束されているわけであるから、１ブロック分の符号化データは毎クロックサイクル当たり４ビット以上の符号語を処理するわけであるから、６４／４＝１６となり、最大でも１６クロックサイクルで着目ブロックの復号処理を終えることになる。すなわち、本復号部１０４を外部から見た場合、目標スループットＴ＝４画素/クロックサイクルを満たす処理を行なうことが可能となる。

【００３８】

図２のテーブルにおいて、“＊”のビットは０、１、のいずれでも良くなり、アドレスのエントリ数は無視できなく大きい。また、各アドレスに対して格納される連結シンボルのビット数も最大で６３ビット必要になるので、そのテーブルを構築するメモリサイズは大きなものは必要になる。

【００３９】

そこで、本第１の実施形態では、装置に必要なメモリ容量を更に削減する例を説明する。

【００４０】

図３は実施形態における復号部１０４のブロック構成図である。

【００４１】
図示において、３００はセレクタ１０３からの符号データを入力する入力端子であり、３０１は符号データから連結符号語を頭出しするためのシフタである。３０２は連結符号語をデコードして、後述のデコードテーブル３０３にアドレスを出力する為のデコーダであり、３０３は図２のデコードテーブルを実現する為のテーブルである。３０４はデコードされた連結シンボルデータを１６ビット単位でパックする為のパッカであり、３０５は１６ビットパック後の画素データを出力する為の出力端子である。また、３０６は連結符号語をデコードして、後述のテーブル３０７及びテーブル３０８にアドレスと供給する為のデコーダであり、テーブル３０７はデコードされた画素数を示すデータ長（以下、パック長と称す）を算出する為のオフセット値（以下、ベース値と称す）を出力する為のテーブルであり、テーブル３０８は符号データの頭だしをする為のシフト量（以下、シフト量と称す）を出力する為のテーブルである。また、３０９はテーブル３０７からのベース値と複数符号データからパック長を算出する為の演算部であり、３１０はブロックの最初の画素値（初期値）を入力する為の１ビットの入力端子である。
【００４２】
次に図３の復号装置の動作を説明する。先ず入力端子３００から入力された符号データはシフタ３０１を介してデコーダ３０２、３０６に供給される。ここで、図２の例では連結符号語の最大ビット数は１１ビットである為、１１ビット分の符号データがデコーダ３０２、３０６に供給される。デコーダ３０２では図２の連結符号語のデコードを行い、デコード結果をテーブルのアドレスとしてテーブル３０３に出力する。ここで、テーブル３０３は図２に示した連結符号語の数としては２１個必要であるが、後述するように本第１の実施形態ではテーブルのエントリー数を５個に減らして、デコーダ３０２及びテーブル３０３の回路規模の削減を図っている。その後、テーブル３０３はデコーダ３０２からのアドレスに従って連結シンボルデータをパッカ３０４に出力する。ここで、後述するようにテーブル３０３から出力する連結シンボルデータは、最後の変化点（２つのランの境界ビット位置）の位置と次画素の値がわかれば良いので、最後の変化点の次画素まで格納されている。即ち、図２の符号語番号“１５”乃至“１９”に対応する連結シンボルデータの時が最大のビット幅となり、このときの最後の変化点の次画素である５ビットのバス幅でパッカ３０４に対して出力される。
【００４３】
一方、デコーダ３０６は、入力された符号データから図２の連結符号語のデコードを行い、デコード結果をテーブルのアドレスとしてテーブル３０７及びテーブル３０８に出力する。ここで、テーブル３０７のエントリー数は、通常、図２に示した連結符号語の数と同等の数が必要であるが、後述するように本第１の実施形態では１０個に減らして、デコーダ３０６、テーブル３０７及びテーブル３０８の回路規模の削減を図っている。
【００４４】
また、図５で説明するように、シフト量の算出に連結符号語の所定のビットを用いる為、演算部３０９に、シフタ３０１から出力された連結符号語が供給されている。その後、テーブル３０７はデコーダ３０６からのアドレスに従ってベース値を演算部３０９へ出力する。演算部３０９では後述するようにベース値と、連結符号語の所定のビット位置の値の総和からパック長を算出し、パッカ３０４に出力する。また、テーブル３０８はデコーダ３０６からのアドレスに従って、シフト量をシフタ３０１に出力する。そして、パッカ３０４ではパック長と、入力端子３１０から入力されるブロックの初期値に従って、連結シンボルデータの内、１番最後の符号語の単体シンボルデータ（以下、最終単体シンボルデータと称す）の生成、及び、出力端子３０５に対する１６ビットにパックした画素データの出力を行なう。又、シフタ３０１はシフト量に従って、次の連結符号語の頭出しを行なう。以上の繰り返しにより、デコード動作を行なう。
【００４５】
次に図４を用いてデコーダ３０２及びテーブル３０３の動作を説明する。符号番号、連結符号語に関しては、図２と同一である。相違点は、デコーダ３０２のアドレス値が追加

されている点と、連結シンボルデータが、４ビット目まで拡張されている点である。例えば、図２の符号語番号０の連結シンボルデータは“１０”であったが、図４では最終単体シンボルのデコード値である２つ目のシンボルのデコード値“０”が拡張されて、連結シンボルデータが“１００００”となっている。これは、後段のパッカ３０４にて、デコードされた画素数を示すパック長を用いて連結シンボルデータのビット幅を決定し、最終単体シンボルのデコード値の開始位置がわかれば、連結シンボルデータの値を決定することが可能である為、最終単体シンボルのデコード値の開始位置が最も長い連結符号語（符号語番号１５乃至１９）の５ビットにバス幅を合わせているためである。つまり、本実施形態で用いている符号化がランレングス符号化であるため、初期値と変化点が分かればデコード可能であり、また、最終変化点以降は同じ値が連続するため、最終変化点、即ち最終単体シンボルのデコード値の開始位置が最も遠くなる位置を含むビット列をテーブルとして保持し、最終ビット以降は最終ビットと同じ値が連続しているものとして取り扱えばよい。このような操作により、連結シンボルデータが同一になるものに対して、デコーダ３０２の出力のアドレスを割り振ることにより、図２にて元々必要であったエントリー数２１個を５個に削減することが可能となる。これにより、デコーダ３０２とテーブル３０３の回路規模の削減を図っている。

【００４６】

次に図５を用いてデコーダ３０６、テーブル３０７及びテーブル３０８の動作を説明する。先ず、単体符号語を識別する為に、各単体符号語の先頭１又は２ビット（以下、ヘッダと称する）を以下の様に定義する。

単体符号語“００”及び“０１”のヘッダは“０”、
単体符号語“１００”及び“１０１”のヘッダは“１０”、
単体符号語“１１＊＊＊＊＊＊”(ここで、“＊＊＊＊＊＊”は０００１００乃至１１１１１１（十進数で４乃至６３の値)のヘッダは“１１”、
単体符号語“１１０００００”（ＥＯＢ）のヘッダは無し。

【００４７】

デコーダ３０６は上記ヘッダに着目して、連結符号語のデコードを行なう。図５はデコーダ３０６の出力であるアドレス毎に図４のデコードテーブルを並び替えた表である。又、“シフト量”はテーブル３０７の出力であるシフト量を示し、“ベース値”はテーブル３０７の出力であり、シフト量を計算する為の各アドレス毎に共通なオフセット値を示している。また“１の数”は各連結符号語のヘッダ以外のビットが“１”になっている数を示す。ここで説明のために符号番号を付与しているが、これは図４のものとは異なる。

【００４８】

次にアドレス及びシフト量のデコード方法とパック長の計算方法に関して図５を用いて説明する。

【００４９】

先ず、符号番号０乃至３に関しては、連結符号語を構成する単体符号語のヘッダが全て“０”である為、“０＊０＊”(＊は０又は１)の連結符号語としてデコードを行い、デコーダ３０６はアドレス“０”を出力する。この時、連結符号語の符号長は“４”である為、シフタ３０１へシフト量“４”を出力する。また、パック長に関してはベースを“２”として、１の数との和により算出する。即ち、符号語番号０の場合は、１の数が０である為、パック長は２＋０＝２となり、符号語番号１,２の場合は、１の数が１である為、パック長は２＋１＝３となる。また、符号語番号３の場合は、１の数が２である為、パック長は２＋２＝４。つまり、連結符号語をcode[0:10]（符号データの先頭を0ビット目とする）とすると、パック長=2+code[1]+code[3]で算出され、パッカ３０４へ出力される。

【００５０】

以下同様に、符号番号４乃至７に関しては、連結符号語を構成する単体符号語のヘッダが同一である為、“０＊１０”（＊は０又は１）の連結符号語としてデコードを行い、デコーダ３０６はアドレス“１”を出力する。この時、連結符号語の符号長は“５”である為、シフタ３０１へシフト量“５”を出力する。また、パック長に関してはベース“４”

＋“１の数”として、パック長=4+code[1]+code[4]で算出され、パッカ３０４へ出力される。

【００５１】

符号番号８は、連結符号語の最後の単体符号語がＥＯＢの場合であり、“０＊１１００００”（＊は０又は１）の連結符号語としてデコードを行い、デコーダ３０６はアドレス“２”を出力する。この時、連結符号語の符号長は“８”である為、シフタ３０１へシフト量“８”を出力する。また、パック長としてはブロックエンドまで（EOB）を示す“６４”が、パッカ３０４へ出力される。

【００５２】

符号番号９、１０に関しては、連結符号語を構成する単体符号語のヘッダが同一である為、“０＊１１＃＃＃＃＃＃”（＊は０又は１であり、＃＃＃＃＃＃は０００１００乃至１１１１１１(十進数の４乃至６２）であり、この値をnとする)の連結符号語としてデコードを行い、デコーダ３０６はアドレス“３”を出力する。この時、連結符号語の符号長は“１０”である為、シフタ３０１へシフト量“１０”を出力する。また、パック長に関してはベース“２”+ “１の数”＋ｎとして、パック長=2+code[1]+nで算出され、パッカ３０４へ出力される。

【００５３】

符号番号１１乃至１４に関しては、連結符号語を構成する単体符号語のヘッダが同一である為、“１０＊０＊”（＊は０又は１)の連結符号語としてデコードを行い、デコーダ３０６はアドレス“４”を出力する。この時、連結符号語の符号長は“５”である為、シフタ３０１へシフト量“５”を出力する。また、パック長に関してはベース“４”＋“１の数”として、パック長=4+code[2]+code[4]で算出され、パッカ３０４へ出力される。

【００５４】

符号番号１５乃至１８に関しては、連結符号語を構成する単体符号語のヘッダが同一である為、“１０＊１０＊”(＊は０又は１)の連結符号語としてデコードを行い、デコーダ３０６はアドレス“５”を出力する。この時、連結符号語の符号長は“６”である為、シフタ３０１へシフト量“６”を出力する。また、パック長に関してはベース“６”+ “１の数”として、パック長=6+code[2]+code[5]で算出され、パッカ３０４へ出力される。

【００５５】

符号番号１９は符号番号２０、２１に関して連結符号語の最後の単体符号語がEOBの場合であり、“１０＊１１００００”（＊は０又は１）の連結符号語としてデコードを行い、デコーダ３０６はアドレス“６”を出力する。この時、連結符号語の符号長は“９”である為、シフタ３０１へシフト量“９”を出力する。また、パック長としてはブロックエンドまで（EOB）を示す“６４”が、パッカ３０４へ出力される。

【００５６】

符号番号２０、２１に関しては、連結符号語を構成する単体符号語のヘッダが同一である為、“１０＊１１＃＃＃＃＃＃”(*は０又は１であり、＃＃＃＃＃＃は０００１００乃至１１１１００（十進数で４乃至６０）であり、この値をｎとする）の連結符号語としてデコードを行い、デコーダ３０６はアドレス“７”を出力する。この時、連結符号語の符号長は“１１”である為、シフタ３０１へシフト量“１１”を出力する。また、パック長に関してはベース“４”+ “１の数”＋ｎとして、パック長=4+code[2] +nで算出され、パッカ３０４へ出力される。

【００５７】

符号番号２２に関しては、連結符号語は単体符号語がEOBそのものである為、そのままデコードを行い、デコーダ３０６はアドレス“８”を出力する。この時、連結符号語の符号長は“６”である為、シフタ３０１へシフト量“６”を出力する。また、パック長としては“６４”が、パッカ３０４へ出力される。

【００５８】

符号番号２３に関しては、連結符号語は単体符号“１１＊＊＊＊＊＊”（＊＊＊＊＊＊は０００１００乃至１１１１１１（十進数で４乃至６３）の値であり、この値をｎとする

）そのものである為、そのままデコードを行い、デコーダ３０６はアドレス“９”を出力する。この時、連結符号語の符号長は“８”である為、シフタ３０１へシフト量“８”を出力する。また、パック長に関してはベース“１”として、パック長=1+nで算出され、パッカ３０４へ出力される。

【００５９】

以上説明したように、デコーダ３０６、テーブル３０７及びテーブル３０８において、連結符号語のデコードをヘッダ構成に着目してデコードし、かつパック長を連結符号語のヘッダ以外のビットの１の数を用いて算出することにより、２４個のエントリー数を１０個に減らす事が可能となり、デコーダ３０６、テーブル３０７及びテーブル３０８の回路規模の削減が可能となる。

【００６０】

なお、上記実施形態ではＥＯＢを検出した場合、パック長＝６４とし、シフタ３０１は固定長端（即ちブロックエンド）まで読み飛ばすものとし、パッカ３０４はテーブル３０３のビット４（最後の画素値）をブロックエンドまで継続するものとして動作する。また、上記実施形態ではＥＯＢのコードそのものを検出しているが、単体符号“１１＊＊＊＊＊＊”として復号し、ｎ＜４となった場合をＥＯＢとしても良い。

【００６１】

次に図６を用いてパッカ３０４の動作を説明する。ここで、パッカ３０４は１ブロック６４ビット分のバッファとカウンタ等の制御回路で構成され、テーブル３０３からの連結シンボルデータを１６ビット単位に出力端子３０５に出力する。

【００６２】

図６の表の行（縦軸）は処理サイクル番号を示し、列（横軸）の項目についているサフィックスｎは各信号のｎサイクル目の状態を示す。例えば、“$sdata_n$”はサイクルｎでテーブル３０３入力される連結シンボルデータである。“end_n”は一つ前のサイクルｎ－１で入力された連結シンボルデータ$sdata_{n-1}$の最終ビット位置の値であり、以下の式（３）で値が決まる。但し、１ブロックの最初の数シンボルデータに関しては、一つ前のサイクルの連結シンボルデータの最終ビット位置の値として、入力端子３１０から与えられた初期値の値を用いて設定される。即ち、初期値が“０”の場合は、一つ前のサイクルの連結シンボルデータが“１”で終わったものと考え、最終ビット位置を“１”に設定する。また、初期値が“１”の場合は、ひとつ前のサイクルの連結シンボルデータが“０”で終わったものと考え、最終ビット位置を“１”に設定する。

$end_n=sdata_{n-1}(4)$ …（３）

【００６３】

また、図６の“sum_n”はサイクルｎまでに１ブロックの処理内で入力されたパック長の和であり、以下の式（４）及び式（５）にて算出される。但し、以下の式（６）に示すように、sum_nが“６４”より大きい場合は次のサイクルで値は保持される。また、後述するサイクルｎまでに出力された１６ビット単位のデータ数（以下、出力パック数と称す）$ocnt_n$が“４”の時、値が“０”に設定される。

sum_{n-1}≦64 かつ $ocnt_n$≠4の時 $sum_n = sum_{n-1} + plen_n$ …（４）

sum_{n-1}＞64 かつ ocntn≠4の時 $sum_n = sum_{n-1}$ …（５）

$ocnt_n$=4の時 $sum_n = 0$ …（６）

【００６４】

図６の“$sptr_n$”は次のサイクルn+1において、連結シンボルデータ$sdata_n$を前記バッファに格納する際の先頭ビット位置を示し、以下の式（７）により算出される。但し、以下の式（７－１）、式（８）に示すようにsum_n＞64の時は$sptr_n$を一つ前のサイクルn-1の値$sptr_{n-1}$にクリップする。後述する出力パック数$ocnt_n$が“４”の時、“０”に設定される。

sum_n≦64 かつ $ocnt_n$≠4の時 $sptr_n = sum_n$-1 …（７）

sum_n＞64 かつ $ocnt_n$≠4の時 $sptr_n = sptr_{n-1}$ …（７－１）

$ocnt_n$= 4の時 $sptr_n = 0$ …（８）

【００６５】
図６の“$eptr_n$”は次のサイクルn+1において、連結シンボルデータ$sdata_n$を前記バッファに格納する際の終了ビット位置を示し、以下の式（９）により算出される。但し、以下の式（１０）、（１１）に示すように、sum_n＞64の場合は前記バッファの最大ビット位置である“６３”に値がクリップされ、後述する出力パック数$ocnt_n$が“４”の時、“０”に設定される。

sum_n≦64 かつ $ocnt_n$≠4の時　$eptr_n = sum_n - 1$　…（９）
sum_n＞64 かつ $ocnt_n$≠4の時　$eptr_n = 63$　…（１０）
$ocnt_n$= 4の時　$eptr_n = 0$　…（１１）

【００６６】
図６の“ext_n”はパック長$plen_n$が連結シンボルデータ$sdata_n$のデータ長5ビットを超える時“１”となるフラグであり、以下の式（１２）、（１３）にて算出される。

$plen_n$ ≦5の時　ext_n= 0　…（１２）
$plen_n$ ＞5の時　ext_n= 1　…（１３）

【００６７】
図６の“$extptr_n$”はパック長$plen_n$が連結シンボルデータ$sdata_n$のデータ長5ビットを超える場合(ext_n= 1の時)、最終ビット位置の値end_nを超えたビット数だけ拡張する必要があるため、次のサイクルn+1において、連結シンボルデータ$sdata_n$を前記バッファに格納する際のビット拡張の開始位置を示す。よって以下の式（１４）、（１５）に示すように、ext_n= 1の時にsum_nから前記越えたビット数分だけ減算することにより算出される。また、以下の式（１６）に示すように、後述する出力パック数$ocnt_n$が“４”の時、値が“０”に設定される。

ext_n=0 かつ$ocnt_n$≠4の時　$extptr_n$= sum_n　…（１４）
ext_n =1かつ$ocnt_n$≠4の時　$extptr_n = sum_n - (plen_n - 5)$　…（１５）
$ocnt_n$=4の時　$extptr_n = 0$　…（１６）

【００６８】
図６の“$pcnt_n$”はサイクルnにおいて、１ブロックの処理内で入力されたパック長の累計が何個のパック数に相当するか（以下、入力パック数と称す）を示し、以下の式（１７）で算出される。ここで、式（１７）の割り算は、小数点以下ビットは切り捨てて演算する。以下の式（１８）に示すように、sum_n＞64の場合は“４”に値がクリップされ、以下の式（１９）に示すように、後述する出力パック数ocntnが“４”の時、“０”に設定される。

sum_n ≦64 かつ $ocnt_n$ ≠4の時　$pcnt_n = sum_n /16$　…（１７）
sum_n ＞64 かつ $ocnt_n$ ≠4の時　$pcnt_n = 4$　…（１８）
$ocnt_n$ = 4の時　$pcnt_n = 0$　…（１９）

【００６９】
図６の“$busy_n$”は図３に図示していない信号であり、サイクルn-1において、次のサイクルnにて前記バッファに空き無くなると判断した場合、テーブル３０３及び演算部３０９に対して、連結シンボルデータ及びパック長の入力を停止させる為の停止信号である。パック長の和sum_{n-1}が６４以上の場合、サイクルnでは前記バッファに空きが無くなる為、１ブロック分のパックデータの出力が終了するまで、値“１”をテーブル３０３及び演算部３０９に対して出力する。よって、以下の式（２０）、（２１）で算出される。また、式（２２）、（２３）、（２４）、（２５）に示すようにテーブル３０３及び演算部３０９は、サイクルnでの連結シンボルデータ$sdata_n$ 及び、パック長$plen_n$はサイクルn-1での$busy_{n-1}$の値が“１”の時にサイクルn-1の値を保持し、“０”の時は値を変更が可能となる。

sum_{n-1}≦64の時　$busy_{n-1} = 0$　…（２０）
sum_{n-1}＞64の時　$busy_{n-1} = 1$　…（２１）
$busy_{n-1}$= １ の時　$sdata_n = sdata_{n-1}$　…（２２）

$busy_{n-1}$= 0 の時 $sdata_n$ は変更可能 …（２３）
$busy_{n-1}$= 1 の時 $plen_n = plen_{n-1}$ …（２４）
$busy_{n-1}$= 0 の時 $plen_n$ は変更可能 …（２５）

【００７０】

図６の“$rest_n$”はサイクルnにおいて、前記入力パック数に対して出力されていないデータのパック数（以下、未出力パック数と称す）を示し、以下の式（２６）に示すように入力パック数$pcnt_n$と、後述する出力パック数$ocnt_n$の差分となる。但し、以下の式（２７）に示すように、後述する出力パック数$ocnt_n$が“４”の時、“０”に設定される。

$ocnt_n \neq 4$の時 $rest_n = pcnt_n - ocnt_n$ …（２６）
$ocnt_n = 4$の時 $rest_n = 0$ …（２７）

【００７１】

図６の“$vaild_n$”は図３に図示していない信号であり、値を後述するパッカ３０４の出力が出力端子３０５に対して有効データであることを示す有効信号であり、“１”の時有効、“０”の時無効を示す。ここで、一つ前のサイクルn-1において、出力されていないパック数“$rest_{n-1}$が“０”より大きければ、パックされたデータを出力することが可能である為、以下の式（２８）に示すように有効信号$vaild_n$の値が“１”に設定される。また、“$rest_{n-1}$の値が“０”であれば、以下の式（２９）に示すように有効信号$vaild_n$の値が“０”に設定される。

$rest_{n-1} > 0$ の時 $vaild_n = 1$ …（２８）
$rest_{n-1} = 0$ の時 $vaild_n = 0$ …（２９）

【００７２】

図6の“$ocnt_n$”は、サイクルnにおける出力パック数を示し、有効信号$vaild_n$と同様に、前のサイクルn+1にて出力されていないパック数“$rest_{n-1}$”が“０”より大きければ、“１”カウントアップを行い、“$rest_{n-1}$”の値が“０”であれば、値を保持する。以下の式（３０）、（３１）にその動作を示す。但し、前のサイクルの出力パック数$ocnt_{n-1}$が“４”の時、以下の式（３２）に示すように、“０”に値をクリアされる。

$rest_{n-1} > 0$ かつ $ocnt_{n-1} \neq 4$の時 $ocnt_n = ocnt_{n-1} + 1$ …（３０）
$rest_{n-1} = 0$ かつ $ocnt_{n-1} \neq 4$の時 $ocnt_n = ocnt_{n-1}$ …（３１）
$ocnt_{n-1} = 4$ の時 $ocnt_n = 0$ …（３２）

【００７３】

図６の“buf_n”は、サイクルn-1における前記１ブロック64ビット分のバッファの値を示し、図６では、パックされる単位である１６ビットづつ分けて示している。“buf_n”の０乃至１５ビット目は“$buf_n(0:15)$”と表わす。また、“$buf_n(16:31)$”は“buf_n”の１６乃至３１ビット目を示し、“$buf_n(32:47)$” は“buf_n”の３２乃至４７ビット目を示す。そして、“$buf_n(48:63)$”は“buf_n”の４８乃至６３ビット目を示す。ここで、サイクルnにおいて、バッファbuf_nに連結シンボルデータ$sdata_{n-1}$を格納する場合、最終ビット位置の値end_{n-1}により、バッファbuf_nに格納する値をビット反転させて格納する。

【００７４】

また、サイクルn-1においてフラグext_{n-1}が“０”の時は、最終ビット位置の値end_{n-1}がビット拡張されずに、データ幅$plen_{n-1}$の連結シンボルデータ$sdata_{n-1}(0: plen_{n-1}-1)$が次のサイクルnにてバッファ$buf_n$に格納される。このとき、連結シンボルデータ$sdata_{n-1}$は先頭ビット位置$sptr_{n-1}$から終了ビット位置$eptr_{n-1}$の領域に格納される為、最終ビット位置の値$end_{n-1}$ が“０”の時は以下の式（３３）、“１”の時は式（３４）となる。ここで“not()”は各ビットのビット反転を示す。

end_{n-1}=0 かつ ext_{n-1}=0の時 $buf_n(sptr_{n-1}: eptr_{n-1}) = sdata_{n-1}(0: plen_{n-1}-1)$ …（３３）
end_{n-1}=1 かつ ext_{n-1}=0の時 $buf_n(sptr_{n-1}: eptr_{n-1}) = not(sdata_{n-1}(0: plen_{n-1}-1))$ …（３４）

【００７５】

また、サイクルn-1においてフラグext_{n-1}が“１”の時は、データ幅$plen_{n-1}$は“５”よ

り大きい為、連結シンボルデータ$sdata_{n-1}(0:4)$が次のサイクルnにてバッファbuf_nに格納される。このとき、連結シンボルデータ$sdata_{n-1}$は先頭ビット位置$sptr_{n-1}$からビット拡張の開始位置$extptr_{n-1}$の１ビット手前の領域に格納される為、最終ビット位置の値end_{n-1}が“０”の時は以下の式（３５）、“１”の時は式（３６）となる。

end_{n-1}=0 かつ ext_{n-1}=1の時　$buf_n(sptr_{n-1}: extptr_{n-1}-1) = sdata_{n-1}(0:4)$　…（３５）

end_{n-1}=1 かつ ext_{n-1}=1の時　$buf_n(sptr_{n-1}: extptr_{n-1}-1) = not(sdata_{n-1}(0:4))$　…（３６）

【００７６】

同時に、フラグext_{n-1}が“１”の時は、連結シンボルデータ$sdata_{n-1}(4)$の値がビット拡張の開始位置$extptr_{n-1}$から終了ビット位置$eptr_{n-1}$の間に拡張されて、次のサイクルnにてバッファbuf_nに格納される為、最終ビット位置の値end_{n-1}が“０”の時は以下の式（３７）、“１”の時は式（３８）となる。

end_{n-1}=0 かつ ext_{n-1}=1の時:

$buf_n(extptr_{n-1}:eptr_{n-1}) = sdata_{n-1}(4), sdata_{n-1}(4),\cdots,sdata_{n-1}(4)$　…（３７）

end_{n-1}=1 かつ ext_{n-1}=1の時:

$buf_n(extptr_{n-1}:eptr_{n-1}) = not(sdata_{n-1}(4), sdata_{n-1}(4),\cdots,sdata_{n-1}(4))$　…（３８）

【００７７】

また、図６の“出力”はパッカ３０４のパックされてたデータの出力であり、それぞれのサイクルにて、出力パック数$ocnt_n$の値に従って、バッファ“$buf_n(0:15)$”、“$buf_n(16:31)$”、“$buf_n(32:47)$”、“$buf_n(48:63)$”のそれぞれ１６ビットの領域を選択して出力する。以下の式（３９）、（４０、（４１）、（４２）に選択動作を示す。

$ocnt_n$=1の時　出力 = $buf_n(0:15)$　…（３９）

$ocnt_n$=2の時　出力 = $buf_n(16:31)$　…（４０）

$ocnt_n$=3の時　出力 = $buf_n(32:47)$　…（４１）

$ocnt_n$=4の時　出力 = $buf_n(47:63)$　…（４２）

【００７８】

次に、パッカ３０４に、１ブロック６４ビット分の連結シンボルデータが、パック長２、５、７、１７、１９、ＥＯＢ（６４）の順で入力され、かつ入力端子３１０に初期値“１”が設定された場合の動作を例にあげて説明する。

【００７９】

先ず、サイクル0では、リセット動作等の初期化手段により、パック長の和sum_0、先頭ビット位置$sptr_0$、終了ビット位置$eptr_0$、フラグext_0、ビット拡張の開始位置$extptr_0$、入力パック数$pcnt_0$、未出力パック数$rest_0$、出力パック数$ocnt_0$に“０”が設定される。また、入力端子３１０より入力されたブロックの最初の画素値が“１”であるので、前のサイクルの連結シンボルデータの最終ビット（即ち、直前の画素値）が“０”であったとみなし最終ビット位置の値end_0に“０”が設定される。このとき、有効信号$vaild_0$は停止信号$busy_0$も初期化されてともに“０”を出力する。

【００８０】

次にサイクル１では、n=1であり、$busy_0$= 0より、式（２３）の条件が成り立ち、連結シンボルデータ$sdata_1$として図５の符号番号０の値“10000”が入力される。同様に$busy_0$ = 0より式（２５）の条件が成り立ち、パック長$plen_1$として“２”が入力される。

【００８１】

また、最終ビット位置の値end_1は式（３）より通常はサイクル０で入力された連結シンボルデータの４ビット目を格納するが、サイクル０では入力されていないので、end_0の値“０”を保持する。

【００８２】

同様に、サイクル０にて連結シンボルデータが入力されていない為、式（３３）乃至（

３８）は成り立たない為、バッファbuf_1へのデータの格納は行なわれない。

【００８３】

出力パック数$ocnt_1$は$rest_0$=0かつ$ocnt_0$=0より式（３１）の条件が成り立ち“０”($ocnt_1$= $ocnt_0$=0)に設定される。

【００８４】

パック長の和sum_1はsum_0 =0かつ$ocnt_1$=0より、式（４）の条件が成り立つ為“２”（sum_1 = sum_0 + $plen_1$ = 0+2 = 2）に設定される。

【００８５】

先頭ビット位置$sptr_1$はsum_1 =2かつ$ocnt_1$=0 より、式（７）の条件が成り立つ為、“０”($sptr_1$ = sum_0 = 0)に設定される。

【００８６】

終了ビット位置$eptr_1$はsum_1=2かつ$ocnt_1$=0 より、式（９）の条件が成り立つ為、“１”($eptr_1$ = sum_1 - 1=2-1=1)に設定される。

【００８７】

フラグext_1はパック長$plen_1$ =2より、式（１２）の条件が成り立つ為、“０”に設定される。

【００８８】

ビット拡張の開始位置$extptr_1$はext_1=0かつ$ocnt_1$=0より式（１４）の条件が成り立つ為、“２”($extptr_1$= sum_1=2)に設定される。

【００８９】

入力パック数$pcnt_1$はsum_1 =2かつ$ocnt_1$=0 より、式（１７）の条件が成り立つ為“０”($pcnt_1$ = sum_1 /16 = 2/16=0)が設定される。

【００９０】

未出力パック数$rest_1$は$ocnt_1$=0より式（２６）の条件が成り立つ為、“０”($rest_1$ = $pcnt_1$ - $ocnt_1$=0-0=0)に設定される。

【００９１】

$valid_1$は$rest_0$=0より式（２９）の条件が成り立ち、“０”が出力される。

【００９２】

さらに、停止信号$busy_1$はsum_1=2より式（２０）の条件が成り立ち“０”が出力される。

【００９３】

次にサイクル２では、n=2であり、$busy_1$= 0より、式（２３）の条件が成り立ち、連結シンボルデータ$sdata_1$として図５の符号番号５の値“10000”が入力される。同様に$busy_2$ = 0より式（２５）の条件が成り立ち、パック長$plen_2$として“５”が入力される。

【００９４】

また、最終ビット位置の値end_2は式（３）より連結シンボルデータ$sdata_1$(4)の値“0”が入力される。

【００９５】

さらに、バッファbuf_2にはend_1=0かつext_1=0であり、式（３３）の条件が成り立つ。式（３３）において、$sptr_1$=0、$eptr_1$=1、$plen_1$=2より、式（３８）の左辺は、buf_2($sptr_1$: $etptr_1$) = buf_2(0:1)となり、右辺は$sdata_1$(0: $plen_1$-1)=$sdata_1$(0: 2-1)= $sdata_1$(0:1) =“１０”となる為、buf_2(0:1)=“１０”となり、バッファbufの０～１ビット目に“１０”が格納される。

【００９６】

また、出力パック数$ocnt_2$は$rest_1$=0かつ$ocnt_1$=0より式（３１）の条件が成り立ち、“０”($ocnt_2$= $ocnt_1$=0)に設定される。

【００９７】

パック長の和sum_2はsum_1 =2かつ$ocnt_2$=0より式（４）の条件が成り立つ為、“７”（sum_2 = sum_1 + $plen_2$ = 2+5 = 7）に設定される。

【００９８】

先頭ビット位置$sptr_2$はsum_2 =7かつ$ocnt_2$=0 より式（７）の条件が成り立つ為“２”($sptr_2 = sum_1 = 2$)に設定される。

【００９９】

終了ビット位置$eptr_2$はsum_2=7かつ$ocnt_2$=0 より式（９）の条件が成り立つ為“６”($eptr_2 = sum_2 - 1=7-1=6$)に設定される。

【０１００】

フラグext_2はパック長$plen_2$ =5より式（１２）の条件が成り立つ為“０”に設定される。

【０１０１】

ビット拡張の開始位置$extptr_2$はext_2=0かつ$ocnt_2$=0より式（１４）の条件が成り立つ為“７”($extptr_2= sum_2$=7)に設定される。

【０１０２】

入力パック数$pcnt_2$はsum_2 =7かつ$ocnt_2$=0 より式（１７）の条件が成り立つ為、“０”($pcnt_2 = sum_2/16 = 7/16=0$)が設定される。

【０１０３】

未出力パック数$rest_2$は$ocnt_2$=0より式（２６）の条件が成り立つ為、“０”($rest_2 = pcnt_2 - ocnt_2=0-0=0$)に設定される。

【０１０４】

有効信号$valid_2$は$rest_1$=0より式（２９）の条件が成り立ち、“０”が出力される。

【０１０５】

さらに、停止信号$busy_2$はsum_2=7より式（２０）の条件が成り立ち“０”が出力される。

【０１０６】

次にサイクル３では、n=3であり、$busy_2$= 0より、式（２３）の条件が成り立ち、連結シンボルデータ$sdata_3$として図５の符号番号１６の値“11100”が入力される。同様に$busy_2$= 0より式（２５）の条件が成り立ち、パック長$plen_3$として“７”が入力される。

【０１０７】

また、最終ビット位置の値end_3は式（３）より連結シンボルデータ$sdata_2(4)$の値“０”が入力される。

【０１０８】

さらに、バッファbuf_3にはend_2=0かつext_2=0であり、式（３３）の条件が成り立つ。式（３３）において、$sptr_2$=2、$eptr_2$=6、$plen_2$=5より、式（３３）の左辺は、$buf_3(sptr_2:etptr_2) = buf_3(2:6)$となる。そして、右辺は$sdata_2(0: plen_2-1)=sdata_2(0: 4)= sdata_2(0:4)$=“10000”となる為、$buf_2(2:6)$=“10000”となり、バッファbufの2～6ビット目に“10000”が格納される。よって、サイクル2で格納されたものと合わせて、$buf_3(0:6)$=“1010000”となる。

【０１０９】

また、出力パック数$ocnt_3$は$rest_2$=0かつ$ocnt_2$=0より式（３１）の条件が成り立ち、“０”($ocnt_3= ocnt_2$=0)に設定される。

【０１１０】

パック長の和sum_3はsum_2 =7かつ$ocnt_3$=0より式（４）の条件が成り立つ為“１４”（$sum_3 = sum_2 + plen_3 = 7+7 = 14$）に設定される。

【０１１１】

先頭ビット位置$sptr_3$はsum_3 =14かつ$ocnt_3$=0 より式（７）の条件が成り立つ為、“７”($sptr_3 = sum_2 = 7$)に設定される。

【０１１２】

終了ビット位置$eptr_3$はsum_3=14かつ$ocnt_3$=0 より式（９）の条件が成り立つ為“１３”($eptr_3 = sum_3 - 1=14-1=13$)に設定される。

【０１１３】

フラグext_3はパック長$plen_3$ =7より式（１３）の条件が成り立つ為“１”に設定される

。
【０１１４】
ビット拡張の開始位置$extptr_3$はext_3=1かつ$ocnt_3$=0より式（１５）の条件が成り立つ為“１２”($extptr_3$= sum_3 - ($plen_3$- 5)=14-(7-5)=12)に設定される。
【０１１５】
入力パック数$pcnt_3$はsum_3 =14かつ$ocnt_3$=0 より式（１７）の条件が成り立つ為“０”($pcnt_3$ = sum_3/16 = 14/16=0)が設定される。
【０１１６】
未出力パック数$rest_3$は$ocnt_3$=0より式（２６）の条件が成り立つ為、“０”($rest_3$ = $pcnt_3$ - $ocnt_3$=0-0=0)に設定される。
【０１１７】
有効信号$valid_3$は$rest_2$=0より式（２９）の条件が成り立ち、“０”が出力される。
【０１１８】
さらに、停止信号$busy_3$はsum_3=14より式（２０）の条件が成り立ち“０”が出力される。
【０１１９】
次にサイクル4では、n=4であり、$busy_3$= 0より、式（２３）の条件が成り立ち、連結シンボルデータ$sdata_4$として図５の符号番号１０の値“11000”が入力される。同様に$busy_3$ = 0より式（２５）の条件が成り立ち、パック長$plen_4$として“１７”が入力される。
【０１２０】
また、最終ビット位置の値end_4は式（３）より連結シンボルデータ$sdata_3$(4)の値“０”が入力される。
【０１２１】
さらに、バッファbuf_4にはend_3=0かつext_3=1であり、式（３５）の条件が成り立つ。式（３５）において、$sptr_3$=7、$extptr_3$=12、$plen_3$=7より、式（３５）の左辺は、buf_4($sptr_3$: $extptr_3$-1) = buf_4(7:12-1)= buf_4(7:11)となり、右辺は$sdata_3$(0:4)=“11100”となる為、buf_4(7:11)=“11100”となる。従ってバッファbufの7～11ビット目に“11100”が格納される。また、式（３７）の条件が成り立つ。式（３７）において、$extptr_3$=12、$eptr_3$=13より、式（３７）の左辺は、buf_4($extptr_3$: $eptr_3$) = buf_4(12:13)となり、右辺は$sdata_3$(4)が2ビット拡張され、“00”となる為、buf_4(12:13)=“00”となる。従って、バッファbufの12～13ビット目に“00”が格納される。よって、サイクル２及び３で格納されたものと合わせて、buf_4(0:13)=“10100001110000”となる。
【０１２２】
また、出力パック数$ocnt_4$は$rest_3$=0かつ$ocnt_3$=0より、式（３１）の条件が成り立ち“０”($ocnt_4$= $ocnt_3$=0)に設定される。
【０１２３】
パック長の和sum_4は、sum_3 =14かつ$ocnt_3$=0より式（４）の条件が成り立つ為、“３１”（sum_4 = sum_3+ $plen_4$ = 14+17 = 31）に設定される。
【０１２４】
先頭ビット位置$sptr_4$はsum_4 =31かつ$ocnt_4$=0 より式（７）の条件が成り立つ為“１４”($sptr_4$= sum_3 = 14)に設定される。
【０１２５】
終了ビット位置$eptr_4$は、sum_4=31かつ$ocnt_4$=0 より式（９）の条件が成り立つ為、“３０”($eptr_4$= sum_4 - 1=31-1=30)に設定される。
【０１２６】
フラグext_4は、パック長$plen_4$ =17より式（１３）の条件が成り立つ為“１”に設定される。
【０１２７】
ビット拡張の開始位置$extptr_4$は、ext_4=1かつ$ocnt_4$=0より式（１５）の条件が成り立つ為、“１９”($extptr_4$= sum_4 - ($plen_4$- 5)=31-(17-5)=19)に設定される。

【０１２８】
入力パック数$pcnt_4$は、sum_4 =31かつ$ocnt_4$=0 より式（１７）の条件が成り立つ為、“１”($pcnt_4 = sum_4/16 = 31/16=1$)が設定される。
【０１２９】
未出力パック数$rest_4$は、$ocnt_4$=0より式（２６）の条件が成り立つ為、“１”($rest_4 = pcnt_4 - ocnt_4=1-0=1$)に設定される。
【０１３０】
有効信号$valid_4$は、$rest_3$=0より式（２９）の条件が成り立ち、“０”が出力される。
【０１３１】
さらに、停止信号$busy_4$は、sum_4=31より式（２０）の条件が成り立ち、“０”が出力される。
【０１３２】
次にサイクル5では、n=5であり、$busy_4$= 0より、式（２３）の条件が成り立ち、連結シンボルデータ$sdata_5$として図５の符号番号２０の値“11100”が入力される。同様に$busy_4$ = 0より式（２５）の条件が成り立ち、パック長$plen_5$として“１９”が入力される。
【０１３３】
また、最終ビット位置の値end_5は式（３）より連結シンボルデータ$sdata_4(4)$の値“０”が入力される。
【０１３４】
さらに、バッファbuf_5にはend_4=0かつext_4=1であり、式（３５）の条件が成り立つ。式（３５）において、$sptr_4$=14、$extptr_4$=19より、式（３５）の左辺は、$buf_5(sptr_4: extptr_4-1) = buf_5(14:19-1)= buf_5(14:18)$となり、右辺は$sdata_4(0:4)$=“11000”となる為、$buf_5(14:18)$=“11000”となる。従って、バッファbufの14～18ビット目に“11000”が格納される。また、式（３７）の条件が成り立つ。式（３７）において、$extptr_4$=19、$eptr_4$=30より式（３７）の左辺は、$buf_5(extptr_4: eptr_4) = buf_5(19:30)$となり、右辺は$sdata_4(4)$が12ビット拡張され、“000000000000”となる為、$buf_5(19:30)$=“000000000000”となる。従って、バッファbufの19～30ビット目に“000000000000”が格納される。よって、サイクル2,3及び4で格納されたものと合わせて、$buf_5(0:15)$=“1010000111000011”及び$buf_5(16:30)$=“000000000000000”となる。
【０１３５】
また、出力パック数$ocnt_5$は、$rest_4$=1かつ$ocnt_4$=0より式（３０）の条件が成り立ち、“１”($ocnt_5= ocnt_4+1=0+1=1$)に設定される。
【０１３６】
パック長の和sum_5は、sum_4 =31かつ$ocnt_5$=1より式（４）の条件が成り立つ為、“５０”（$sum_5= sum_4+ plen_5 = 31+19 = 50$）に設定される。
【０１３７】
先頭ビット位置$sptr_5$は、sum_5 =50かつ$ocnt_5$=1 より式（７）の条件が成り立つ為、“３１”($sptr_5 = sum_4 = 31$)に設定される。
【０１３８】
終了ビット位置$eptr_5$は、sum_5=50かつ$ocnt_5$=1より式（９）の条件が成り立つ為、“４９”($eptr_5= sum_5 - 1=50-1=49$)に設定される。
【０１３９】
フラグext_5は、パック長$plen_5$ =19より式（１３）の条件が成り立つ為“１”に設定される。
【０１４０】
ビット拡張の開始位置$extptr_5$は、ext_5=1かつ$ocnt_5$=1より式（１５）の条件が成り立つ為、“３６”($extptr_5= sum_5 - (plen_5- 5)=50-(19-5)=36$)に設定される。
【０１４１】
入力パック数$pcnt_5$は、sum_5 =50かつ$ocnt_5$=1 より式（１７）の条件が成り立つ為、“３”($pcnt_5 = sum_5/16 = 50/16=3$)が設定される。

【０１４２】
未出力パック数$rest_5$は、$ocnt_5$=1より式（２６）の条件が成り立つ為、“２”($rest_5$ = $pcnt_5$ - $ocnt_5$=3-1=2)に設定される。
【０１４３】
有効信号$valid_5$は、$rest_4$=1より式（２８）の条件が成り立ち、“１”が出力される。このとき、出力パック数$ocnt_5$が“１”である為、式（３９）が成り立ち、パックされた出力データとしてバッファbuf5(0:15)の値“1010000111000011”選択されて出力される。
【０１４４】
さらに、停止信号$busy_5$はsum_5=50より式（２０）)の条件が成り立ち“０”が出力される。
【０１４５】
次にサイクル６では、n=6であり、$busy_5$= 0より、式（２３）の条件が成り立ち、連結シンボルデータ$sdata_6$として図５の符号番号９の連結シンボルデータで、最終単体符号語としてEOB(64)が入力されているものとする。即ち、“10000”が入力される。同様に$busy_5$= 0より式（２５）の条件が成り立ち、パック長$plen_6$として“EOB”すなわち64が入力される。
【０１４６】
また、最終ビット位置の値end_6は式（３）より、連結シンボルデータsdata5(4)の値“０”が入力される。
【０１４７】
さらに、バッファbuf_6にはend_5=0かつext_5=1であり、式（３５）の条件が成り立つ。式（３５）において、$sptr_5$=31、$extptr_5$=36より、式（３５）の左辺は、buf_6($sptr_5$: $extptr_5$-1) = buf_6(31:36-1)= buf_6(31:35)となる。右辺は$sdata_5$(0:4)=“11100”となる為、buf_6(31:35)=“11100”となる。従って、バッファbufの31～35ビット目に“11100”が格納される。また、式（３７）の条件が成り立つ。式（３７）において、$extptr_5$=36、$eptr_5$=49より、式（３７）の左辺は、buf_6($extptr_5$: $eptr_5$) = buf_6(36:49)となる。また右辺の$sdata_5$(4)が14ビット拡張され、“00000000000000”となる為、buf_6(36:49)=“00000000000000”となる。従って、バッファbufの36～49ビット目に“00000000000000”が格納される。よって、サイクル2,3,4及び5で格納されたものと合わせて、buf_6(0:15)=“1010000111000011”, buf_6(16:31)=“0000000000000001”, buf_6(32:47)=“1110000000000001”, 及びbuf_6(48:49)=“00”となる。
【０１４８】
また、出力パック数$ocnt_6$は、$rest_5$=2かつ$ocnt_5$=1より式（３０）の条件が成り立ち、“２”($ocnt_6$= $ocnt_5$+1=1+1=2)に設定される。
【０１４９】
パック長の和sum_6は、sum_5 =50かつ$ocnt_6$=2より式（４）の条件が成り立つ為、“１１４”（sum_6= sum_5+ $plen_6$ = 50+64 = 114）に設定される。
【０１５０】
先頭ビット位置$sptr_6$は、sum_6 =114かつ$ocnt_6$=2 より式（７）の条件が成り立つ為、“５０”($sptr_6$ = sum_5 = 50)に設定される。
【０１５１】
終了ビット位置$eptr_6$は、sum_6=114かつ$ocnt_6$=2より式（１０）の条件が成り立つ為、“６３”に設定される。
【０１５２】
フラグext_6は、パック長$plen_6$ =64より式（１３）の条件が成り立つ為、“１”に設定される。
【０１５３】
ビット拡張の開始位置$extptr_6$は、ext_6=1かつ$ocnt_6$=2より式（１５）の条件が成り立つ為、“５５”($extptr_6$= sum_6 - ($plen_6$- 5)=114-(64-5)=55)に設定される。
【０１５４】

入力パック数$pcnt_6$は、sum_6 =114かつ$ocnt_6$=2 より式（１８）の条件が成り立つ為、“４”に設定される。

【０１５５】

未出力パック数$rest_6$は、$ocnt_6$=2より式（２６）の条件が成り立つ為、“２”($rest_6$ = $pcnt_6$- $ocnt_6$=4-2=2)に設定される。

【０１５６】

有効信号$valid_6$は$r、est_5$=2より式（２８）の条件が成り立ち、“１”が出力される。このとき、出力パック数$ocnt_6$が“２”である為、式（４０）が成り立ち、パックされた出力データとしてバッファbuf_5(16:31)の値“0000000000000001”が選択されて出力される。

【０１５７】

さらに、停止信号$busy_6$は、sum_6=114より式（２１）の条件が成り立ち“１”が出力される。

【０１５８】

次にサイクル７では、n=7であり、$busy_6$= 1より、式（２２）の条件が成り立ち、テーブル３０３に対して連結シンボルデータの入力を停止する（サイクル６のデータを保持させる）。同様に$busy_6$= 0より式（２４）の条件が成り立ち、演算部３０９に対してパック長の入力を停止する（サイクル6のデータを保持させる）。

【０１５９】

また、最終ビット位置の値end_7は式（３）より、連結シンボルデータ$sdata_6$(4)の値“０”が入力される。

【０１６０】

さらに、バッファbuf_7にはend_6=0かつext_6=1であり、式（３５）の条件が成り立つ。式（３５）において、$sptr_6$=50、$extptr_6$=55より、式（３５）の左辺は、buf_7($sptr_7$: $extptr_7$-1) = buf_7(50:55-1)= buf_7(50:54)となる。また、右辺は$sdata_6$(0:4)=“10000”となる為、buf_7(50:54)=“10000”となる。従って、バッファbufの50～54ビット目に“10000”が格納される。また、式（３７）の条件が成り立つ。式（３７）において、$extptr_6$=55、$eptr_6$=63より、式（３７）の左辺は、buf_7($extptr_6$: $eptr_6$) = buf_6(55:63)となる。また、右辺は$sdata_6$(4)が9ビット拡張され、“000000000”となる為、buf_7(55:63)= 000000000”となる。従って、バッファbufの55～63ビット目に000000000”が格納される。よって、サイクル2,3,4,5及び6で格納されたものと合わせて、buf_7(0:15)=“1010000111000011”, buf_7(16:31)=“0000000000000001”, buf_7(32:47)=“1110000000000001”,及びbuf_7(48:63)=“ 0010000000000000”となる。

【０１６１】

また、出力パック数$ocnt_7$は、$rest_6$=2かつ$ocnt_6$=2より式（３０）の条件が成り立ち、“３”($ocnt_7$= $ocnt_6$+1=2+1=3)に設定される。

【０１６２】

パック長の和sum_7は、sum_6 =114かつ$ocnt_7$=3より式（５）の条件が成り立つ為、“１１４”（sum_7= sum_6= 114）に設定される。

【０１６３】

先頭ビット位置$sptr_7$は、sum_7 =114かつ$ocnt_7$=3 より式（７－１）の条件が成り立つ為、“５０”($sptr_7$ = $sptr_6$= 50)に設定される。

【０１６４】

終了ビット位置$eptr_7$は、sum_7=114かつocnt7=3より式（１０）の条件が成り立つ為、“６３”に設定される。

【０１６５】

フラグext_7は、パック長$plen_7$ =64より式（１３）の条件が成り立つ為、“１”に設定される。

【０１６６】

ビット拡張の開始位置$extptr_7$は、ext_7=1かつ$ocnt_7$=3より式（１５）の条件が成り立つ

為、“５５”($extptr_7$= sum_7 - ($plen_7$- 5)=114-(64-5)=55)に設定される。

【０１６７】

入力パック数$pcnt_7$は、sum_7=114かつ$ocnt_7$=3 より式（１８）の条件が成り立つ為“４”に設定される。

【０１６８】

未出力パック数$rest_7$は、$ocnt_7$=3より式（２６）の条件が成り立つ為、“１”($rest_7$ = $pcnt_7$- $ocnt_7$=4-3=1)に設定される。

【０１６９】

有効信号$valid_7$は、$rest_7$=1より式（２８）の条件が成り立ち、“１”が出力される。このとき、出力パック数$ocnt_7$が“３”である為、式（４１）が成り立ち、パックされた出力データとしてバッファbuf_7(32:47)の値“1110000000000001”選択されて出力される。

さらに、停止信号$busy_7$は、sum_7=114より式（２１）の条件が成り立ち、“１”が出力される。

【０１７０】

次にサイクル８では、n=8であり、サイクル７と同様に$busy_7$= 1より式（２２）の条件が成り立ち、テーブル３０３に対して連結シンボルデータの入力を停止する（サイクル７のデータを保持させる）。同様に$busy_7$= 0より式（２４）の条件が成り立ち、演算部３０９に対してパック長の入力を停止する（サイクル７のデータを保持させる）。

【０１７１】

また、最終ビット位置の値end_8は、サイクル７と同様に式（３）より連結シンボルデータ$sdata_7$(4)の値“０”が入力される。

【０１７２】

さらに、バッファbuf_8は、サイクル７において、end_7、ext_7、$sptr_7$、$extptr_7$、$eptr_7$等の値が、サイクル６の時と変化していないため、バッファbuf_8のデータはバッファbuf_7と同等である。

【０１７３】

また、出力パック数$ocnt_8$は、$rest_7$=1かつ$ocnt_7$=3より式（３０）の条件が成り立ち、“４”($ocnt_8$= $ocnt_7$+1=3+1=4)に設定される。

【０１７４】

パック長の和sum_8は、$ocnt_8$=4より式（６）の条件が成り立つ為、“０”に設定される。

【０１７５】

先頭ビット位置$sptr_8$も同様に、$ocnt_8$=4 より式（８）の条件が成り立つ為、“０”設定される。

【０１７６】

終了ビット位置$eptr_7$も同様にo、cnt_8=4より式（１１）の条件が成り立つ為“０”に設定される。

【０１７７】

フラグext_8は、パック長$plen_8$ =64より式（１３）の条件が成り立つ為、“１”に設定される。

【０１７８】

ビット拡張の開始位置$extptr_8$も同様に、$ocnt_8$=4より式（１６）の条件が成り立つ為“０”に設定される。

【０１７９】

入力パック数$pcnt_7$も同様に、ocnt8=4 より式（１９）の条件が成り立つ為“０”に設定される。

【０１８０】

未出力パック数$rest_8$も同様に、$ocnt_8$=4より式（２７）の条件が成り立つ為“0”に設定される。

【０１８１】
有効信号$valid_7$は、$rest_7$=1より式（２８）の条件が成り立ち、“１”が出力される。このとき、出力パック数$ocnt_8$が“４”である為、式（４２）が成り立ち、パックされた出力データとしてバッファbuf_8(48:63)の値“0010000000000000”が選択されて出力される。

【０１８２】
さらに、停止信号$busy_8$は、sum_8=0より式（２０）の条件が成り立ち、“０”が出力される。

【０１８３】
以上のサイクル１～サイクル８の動作により、１ブロック６４ビットのパック動作が完了する。次のサイクルでは、出力パック数$ocnt_9$は、$ocnt_8$=4より式（３２）の条件が成り立ち、“０”にクリアされ、前記サイクル０と同様な、初期状態に移行する。

【０１８４】
［第２の実施形態］
次に、図７Ａ、７Ｂを用いて第２の実施形態のデコードテーブルを説明する。ここで図２と同様に説明のためにそれぞれの連結符号語に対して符号語番号を付与している。

【０１８５】
符号語番号０の連結符号語は、単体符号語“００”が３つ続けてデコードされる場合であり、デコードされた連結シンボルデータは“１０１”（３画素が復号される）となる。また、符号語番号１の連結符号語は単体符号語“００”が２つ続き、その後に単体符号語“０１”が続けてデコードされる場合であり、デコードされた連結シンボルデータは“１０１１”となる。また、符号語番号２の連結符号語は単体符号語“００”が２つ続き、その後に単体符号語“１００”が続けてデコードされる場合であり、デコードされた連結シンボルデータは“１０１１１”となる。また、符号語番号３の連結符号語は単体符号語“００”が２つ続き、その後に単体符号語“１０１”が続けてデコードされる場合であり、デコードされた連結シンボルデータは“１０１１１１”となる。さらに、符号語番号４の連結符号語は単体符号語“００”が２つ続き、その後に単体符号語“１１＊＊＊＊＊＊”(ここで、＊＊＊＊＊＊は000100-111101(十進数で４乃至６１）のいずれかの値)が続けてデコードされる場合であり、デコードされた連結シンボルデータは“１０”の後、５乃至６２個の“１”が続く連結シンボルデータとなる。

【０１８６】
同様な方法で、図７Ａ，７Ｂには連結シンボルデータを構成する４つの連続する単体符号語に対して単体符号語“００”，“０１”，“１００”，“１０１”，“１１＊＊＊＊＊”がそれぞれ出現した場合の組み合わせが列挙されている。但し、符号語番号２０，４１，６２，８３においては、連結符号語は２番目の単体符号語を“１１＊＊＊＊＊＊”とする２つの単体符号語で構成されている。また、符号語番号８４においては、連結符号語は単体符号語“１１＊＊＊＊＊＊”のみで構成されている。

【０１８７】
尚、図示していない例外処理として、単体符号語“ＥＯＢ”が連結符号語の途中で出現した場合は、１ブロックの残りの画素全てに関して、直前にデコードされたシンボルの値が０だった場合は１を連結シンボルデータを出力する。また、直前にデコードされたシンボルの値が１だった場合は０を連結シンボルデータを出力する。さらに、単体符号語“ＥＯＢ”が連結符号語の先頭で出現した場合(連結符号語自身がＥＯＢの場合)は、直前の画素データが０の場合には、１ブロックの残りの画素全てを１として出力する。また、直前の画素データが１の場合は１ブロック残りの画素全て０を出力する。

【０１８８】
図８は、第２の実施形態の復号部１０４のブロック構成図である。図中、８００は符号データの入力端子であり、８０１は符号データから連結符号語を頭出しするためのシフタであり、８０２は連結符号語をデコードして、後述のデコードテーブル８０３にアドレスを出力する為のデコーダである。８０３は図７Ａ，７Ｂのデコードテーブルを実現する為

のテーブルであり、８０４はデコードされた連結シンボルデータを１６ビット単位でパックする為のパッカである。８０５は１６ビットパック後の画素データを出力する為の出力端子である。また、８０６は連結符号語をデコードして、後述のテーブル８０７及びテーブル８０８にアドレスを供給する為のデコーダである。８０７はデコードされた画素数を示すデータ長（以下、パック長と称す）を算出する為のオフセット値（以下、ベース値と称す）を出力する為のテーブルである。８０８は符号データの頭だしをする為のシフト量（以下、シフト量と称す）を出力する為のテーブルである。また、８０９はテーブル８０７からのベース値と複数符号データからパック長を算出する為の演算部であり、８１０はブロックの最初の画素値（初期値）を入力する為の１ビットの入力端子である。

【０１８９】

次に図８の復号部の処理内容を説明する。先ず入力端子８００から入力された符号データはシフタ８０１を介してデコーダ８０２、８０８に入力される。ここで、図７Ａ，７Ｂの例では連結符号語の最大ビット数は１４ビットである為、１４ビット分の符号データがデコーダ８０２、８０６及びテーブル８０７に入力される。デコーダ８０２では図７Ａ，７Ｂの連結符号語のデコードを行い、デコード結果をテーブルのアドレスとしてテーブル８０３に出力する。

【０１９０】

ここで、テーブル８０３は図７Ａ，７Ｂに示した連結符号語の数として８５個必要であるが、後述するように第２の実施形態ではテーブルのエントリー数を２１個に減らして、デコーダ８０２及びテーブル８０３の回路規模の削減を図っている。

【０１９１】

その後、テーブル８０３はデコーダ８０２からのアドレスに従って連結シンボルデータをパッカ８０４に出力する。ここで、後述するようにテーブル８０３から出力する連結シンボルデータは、図３で説明したのと同様に、最後の変化点の位置と次画素の値がわかれば良いので、最後の変化点の次画素まで格納されている。即ち、図７Ａ，７Ｂの符号語番号７８～８２に対応する連結シンボルデータの時、最大のビット幅となり、このときの最後の変化点の次画素である９ビットのバス幅でパッカ８０４に対して出力される。

【０１９２】

一方、デコーダ８０６に入力された符号データは、図７Ａ，７Ｂの連結符号語のデコードを行い、デコード結果をテーブルのアドレスとしてテーブル８０７及びテーブル８０８に出力する。ここで、テーブル８０７のエントリー数は、図７Ａ，７Ｂに示した連結符号語の数と同等の数が必要であるが、後述するように第２の実施形態では２２個に減らして、デコーダ８０６、テーブル８０７及びテーブル８０８の回路規模の削減を図っている。

【０１９３】

また、図１０Ａ，１０Ｂで説明するように、シフト量の算出に連結符号語の所定のビットを用いる為、演算部８０９に、シフタ８０１から出力された連結符号語が入力されている。その後、テーブル８０７はデコーダ８０６からのアドレスに従ってベース値を演算部８０９に出力する。演算部８０９では後述するようにベース値と、連結符号語の所定のビット位置の値の総和からパック長を算出し、パッカ８０４に出力し、かつシフト量をシフタ８０１に出力する。その後、パッカ８０４ではパック長と、入力端子８１０から入力されるブロックの初期値に従って、連結シンボルデータの最終単体シンボルデータの生成、及び、出力端子８０５に対する１６ビットにパックした画素データの出力を行なう。又、シフタ８０１はシフト量に従って、次の連結符号語の頭出しを行なう。以上の繰り返しにより、デコード動作を行なう。

【０１９４】

次に図９Ａ，９Ｂを用いてデコーダ８０２及びテーブル８０３の動作を説明する。符号番号、連結符号語に関しては、図７Ａ，７Ｂと同一である。相違点は、デコーダ８０２のアドレス値が追加されている点と、連結シンボルデータが、８ビット目まで拡張されている点である。例えば、図７Ａ，７Ｂの符号語番号０の連結シンボルデータは“１０１”であったが、図９Ａ，９Ｂでは最終単体シンボルのデコード値である３つ目のシンボルのデ

コード値“１”が拡張されて、連結シンボルデータが“101111111”となっている。これは、後段のパッカ８０４にて、デコードされた画素数を示すパック長を用いて連結シンボルデータのビット幅を決定し、最終単体シンボルのデコード値の開始位置と値がわかれば、連結シンボルデータの値が決定することが可能である為、最終単体シンボルのデコード値の開始位置が最も長い連結符号語（符号語番号７８乃至８２）の９ビットにバス幅を合わせているためである。これらのビット拡張により、連結シンボルデータが同一になるものに対して、デコーダ３０２の出力のアドレスを割り振ることにより、図７Ａ，７Ｂにて元々必要であったエントリー数８５個を２１個に削減することが可能となる。これにより、デコーダ８０２とテーブル８０３の回路規模の削減を図っている。

【０１９５】

次に図１０Ａ，１０Ｂを用いてデコーダ８０６、テーブル８０７及びテーブル８０８の動作を説明する。先ず、単体符号語を識別する為のヘッダの使用は、図５で説明したものと同等である。

【０１９６】

デコーダ８０６は単体符号のヘッダに着目して、連結符号語のデコードを行なう。図１０Ａ，１０Ｂはデコーダ８０６の出力であるアドレス毎に図９Ａ，９Ｂのデコードテーブルを並び替えた表である。又、“シフト量”はテーブル８０７の出力であるシフト量を示し、“ベース値”はシフト量を計算する為の各アドレス毎に共通なオフセット値を示し、“１の数”は各連結符号語のヘッダ以外のビットが“１”になっている数を示す。ここで説明のために符号番号を付与しているが、これは図９Ａ，９Ｂのものとは異なる。

【０１９７】

次にアドレス及びシフト量のデコード方法とパック長の計算方法に関して図１０Ａ，１０Ｂを用いて説明する。

【０１９８】

先ず、符号番号０乃至７に関しては、連結符号語を構成する単体符号語のヘッダが全て“０”である為、“0*0*0*”(*は０又は１)の連結符号語としてデコードを行い、デコーダ８０６はアドレス“０”を出力する。この時、連結符号語の符号長は“６”である為、シフタ８０１へシフト量“６”を出力する。また、パック長に関してはベースを“３”として、１の数との和により算出する。即ち、符号語番号０の時は、１の数が０である為、パック長は3+0=3となる。符号語番号１，２，４の時は、１の数が１つである為、パック長は3+1=4となる。また、符号語番号３，５，６の時は、１の数が２つである為、パック長は3+2=5となる。符号語番号７の時は、１の数が３つである為、パック長は3+3=6となる。つまり、連結符号語をcode[0:13]（符号データの先頭を0ビット目とする）とすると、パック長=3+code[1]+code[3]+code[5]で算出され、パッカ８０４へ出力される。

【０１９９】

以下同様に、符号番号８乃至１５に関しては、連結符号語を構成する単体符号語のヘッダが同一である為、“0*0*0*”(*は０又は１)の連結符号語としてデコードを行い、デコーダ８０６はアドレス“１”を出力する。この時、連結符号語の符号長は“７”である為、シフタ８０１へシフト量“７”を出力する。また、パック長に関してはベース“５”+“１の数”として、パック長=5+code[1]+code[3] +code[6]で算出され、パッカ８０４へ出力される。

【０２００】

符号番号１６は、連結符号語の最後の単体符号語がEOBの場合であり、“0*0*110000”(*は0又は1)の連結符号語としてデコードを行い、デコーダ８０６はアドレス“３”を出力する。この時、連結符号語の符号長は“１０”である為、シフタ８０１へシフト量“１０”を出力する。また、パック長に関してはブロックエンドまで(EOB)を示す“６４”として、パッカ８０４へ出力される。

【０２０１】

符号番号１７乃至２０に関しては、連結符号語を構成する単体符号語のヘッダが同一である為、“0*0*11######”(*は0又は1、######は000100-111101(十進数で4-61)であり、

この数字をnとする)の連結符号語としてデコードを行い、デコーダ８０６はアドレス“３”を出力する。この時、連結符号語の符号長は“１２”である為、シフタ８０１へシフト量“１２”を出力する。また、パック長に関してはベース“３”+“１の数”+nとして、パック長=3+code[1]+code[3]+nで算出され、パッカ８０４へ出力される。

【０２０２】

符号番号２１乃至２８に関しては、連結符号語を構成する単体符号語のヘッダが同一である為、“0*10*0*”(*は0又は1)の連結符号語としてデコードを行い、デコーダ８０６はアドレス“４”を出力する。この時、連結符号語の符号長は“７”である為、シフタ８０１へシフト量“７”を出力する。また、パック長に関してはベース“５”+“１の数”として、パック長=5+code[1]+code[4] +code[6]で算出され、パッカ８０４へ出力される。

【０２０３】

符号番号２９無し３６に関しては、連結符号語を構成する単体符号語のヘッダが同一である為、“0*10*10*”(*は0又は1)の連結符号語としてデコードを行い、デコーダ８０６はアドレス“５”を出力する。この時、連結符号語の符号長は“８”である為、シフタ８０１へシフト量“８”を出力する。また、パック長に関してはベース“７”+“１の数”として、パック長=7+code[1]+code[4] +code[7]で算出され、パッカ８０４へ出力される。

【０２０４】

符号番号３７は符号番号３８乃至４１に関して連結符号語の最後の単体符号語がEOBの場合であり、“0*10*110000”(*は0又は1)の連結符号語としてデコードを行い、デコーダ８０６はアドレス“６”を出力する。この時、連結符号語の符号長は“１１”である為、シフタ８０１へシフト量“１１”を出力する。また、パック長に関してはブロックエンドまで(EOB)を示す“６４”として、パッカ8８０４へ出力される。

【０２０５】

符号番号３８乃至４１に関しては、連結符号語を構成する単体符号語のヘッダが同一である為、“0*10*11######”(*は0又は1、######は000100-111011(十進数で4乃至59)であり、この数字をnとする)の連結符号語としてデコードを行い、デコーダ８０６はアドレス“７”を出力する。この時、連結符号語の符号長は“１３”である為、シフタ８０１へシフト量“１３”を出力する。また、パック長に関してはベース“５”+“１の数”+nとして、パック長=5+code[1]+code[4]+nで算出され、パッカ８０４へ出力される。

【０２０６】

符号番号４２は符号番号４３、４４に関して連結符号語の最後の単体符号語がEOBの場合であり、“0*110000”(*は0又は1)の連結符号語としてデコードを行うことになる。デコーダ８０６はアドレス“８”を出力する。この時、連結符号語の符号長は“８”である為、シフタ８０１へシフト量“８”を出力する。また、パック長に関してはブロックエンドまで(EOB)を示す“６４”として、パッカ８０４へ出力される。

【０２０７】

符号番号４３、４４に関しては、連結符号語を構成する単体符号語のヘッダが同一である為、“0*11######”(*は0又は1、######は000100-111110(十進数で4-62)であり、この数字をnとする)の連結符号語としてデコードを行い、デコーダ８０６はアドレス“９”を出力する。この時、連結符号語の符号長は“１０”である為、シフタ８０１へシフト量“１０”を出力する。また、パック長に関してはベース“２”+“１の数”+nとして、パック長=2+code[1]+nで算出され、パッカ８０４へ出力される。

【０２０８】

符号番号４５乃至５２に関しては、連結符号語を構成する単体符号語のヘッダが同一である為、“10*0*0*”(*は0又は1)の連結符号語としてデコードを行い、デコーダ８０６はアドレス“１０”を出力する。この時、連結符号語の符号長は“７”である為、シフタ８０１へシフト量“７”を出力する。また、パック長に関してはベース“５”+“１の数”として、パック長=5+code[2]+code[4] +code[6]で算出され、パッカ８０４へ出力される。

【０２０９】
符号番号５３乃至６０に関しては、連結符号語を構成する単体符号語のヘッダが同一である為、“10*0*10*”(*は0又は1)の連結符号語としてデコードを行い、デコーダ８０６はアドレス“１１”を出力する。この時、連結符号語の符号長は“８”である為、シフタ８０１へシフト量“８”を出力する。また、パック長に関してはベース“７”+“１の数”として、パック長=7+code[2]+code[4] +code[7]で算出され、パッカ８０４へ出力される。
【０２１０】
符号番号６１は符号番号６２乃至６５に関して連結符号語の最後の単体符号語がEOBの場合であり、“10*0*110000”(*は0又は1)の連結符号語としてデコードを行う。デコーダ８０６はアドレス“１２”を出力する。この時、連結符号語の符号長は“１１”である為、シフタ８０１へシフト量“１１”を出力する。また、パック長に関してはブロックエンドまで(EOB)を示す“６４”となり、パッカ８０４へ出力される。
【０２１１】
符号番号６２乃至６５に関しては、連結符号語を構成する単体符号語のヘッダが同一である為、“10*0*11######”(*は0又は1、######は000100-111011(十進数で4-59)であり、この数字をnとする)の連結符号語としてデコードを行い、デコーダ８０６はアドレス“１３”を出力する。この時、連結符号語の符号長は“１３”である為、シフタ８０１へシフト量“１３”を出力する。また、パック長に関してはベース“５”+“１の数”+nとして、パック長=5+code[2]+code[4]+nで算出され、パッカ８０４へ出力される。
【０２１２】
符号番号６６乃至７３に関しては、連結符号語を構成する単体符号語のヘッダが同一である為、“10*10*0*”(*は0又は1)の連結符号語としてデコードを行い、デコーダ８０６はアドレス“１４”を出力する。この時、連結符号語の符号長は“８”である為、シフタ８０１へシフト量“８”を出力する。また、パック長に関してはベース“７”+“１の数”として、パック長=7+code[2]+code[5] +code[7]で算出され、パッカ８０４へ出力される。
【０２１３】
符号番号７４乃至８１に関しては、連結符号語を構成する単体符号語のヘッダが同一である為、“10*10*10*”(*は0又は1)の連結符号語としてデコードを行い、デコーダ８０６はアドレス“１５”を出力する。この時、連結符号語の符号長は“９”である為、シフタ８０１へシフト量“９”を出力する。また、パック長に関してはベース“９”+“１の数”として、パック長=9+code[2]+code[5] +code[8]で算出され、パッカ８０４へ出力される。
【０２１４】
符号番号８２は符号番号８３乃至８６に関して連結符号語の最後の単体符号語がEOBの場合であり、“10*10*110000”(*は0又は1)の連結符号語としてデコードを行い、デコーダ８０６はアドレス“１６”を出力する。この時、連結符号語の符号長は“１２”である為、シフタ８０１へシフト量“１２”を出力する。また、パック長に関してはブロックエンドまで(EOB)を示す“６４”として、パッカ８０４へ出力される。
【０２１５】
符号番号８３乃至８６に関しては、連結符号語を構成する単体符号語のヘッダが同一である為、“10*10*11######”(*は0又は1、######は000100-111001(十進数で4-57)であり、この数字をnとする)の連結符号語としてデコードを行い、デコーダ８０６はアドレス“１７”を出力する。この時、連結符号語の符号長は“１４”である為、シフタ８０１へシフト量“１４”を出力する。また、パック長に関してはベース“７”+“１の数”+nとして、パック長=7+code[2]+code[5]+nで算出され、パッカ８０４へ出力される。
【０２１６】
符号番号８７は符号番号８８、８９に関して連結符号語の最後の単体符号語がEOBの場合であり、“10*110000”(*は0又は1)の連結符号語としてデコードを行う。デコーダ８０

６はアドレス“１８”を出力する。この時、連結符号語の符号長は“９”である為、シフタ８０１へシフト量“９”を出力する。また、パック長に関してはブロックエンドまで(EOB)を示す“６４”として、パッカ８０４へ出力される。

【０２１７】

符号番号８８乃至８９に関しては、連結符号語を構成する単体符号語のヘッダが同一である為、“10*11######”(*は0又は1、######は000100-111100(十進数で4-60)であり、この数字をnとする)の連結符号語としてデコードを行う。デコーダ８０６はアドレス“１９”を出力する。この時、連結符号語の符号長は“１１”である為、シフタ８０１へシフト量“１１”を出力する。また、パック長に関してはベース“４”+“１の数”+nとして、パック長=4+code[2]+nで算出され、パッカ８０４へ出力される。

【０２１８】

符号番号９０に関しては、連結符号語は単体符号語がEOBそのものである為、そのままデコードを行い、デコーダ８０６はアドレス“２０”を出力する。この時、連結符号語の符号長は“６”である為、シフタ８０１へシフト量“６”を出力する。また、パック長に関してはブロックエンドまで(EOB)を示す“６４”として、パッカ８０４へ出力される。

【０２１９】

符号番号９１に関しては、連結符号語は単体符号“11******”(******は000100-111111(十進数で4-63)の値であり、この数字をnとする)そのものである為、そのままデコードを行う。デコーダ８０６はアドレス“２１”を出力する。この時、連結符号語の符号長は“８”である為、シフタ８０１へシフト量“８”を出力する。また、パック長に関してはベース“１”として、パック長=1+nで算出され、パッカ８０４へ出力される。

【０２２０】

以上説明したように、デコーダ８０６及びテーブル８０７において、連結符号語のデコードをヘッダ構成に着目してデコードし、かつパック長を連結符号語のヘッダ以外のビットの１の数を用いて算出することにより、９２個のエントリー数を２２個に減らす事が可能となり、デコーダ８０６とテーブル８０７の回路規模の削減が可能となる。

【０２２１】

次にパッカ８４の動作を説明する。パッカ８０４はパッカ３０４に対して入力される連結シンボルデータのビット幅が異なり、パッカ３０４の５ビットに対して、パッカ８０４は９ビットであるため、前記の式（３）、（１２）、（１３）、（１５）、（３５）、（３６）、（３７）、及び、（３８）のの演算及び条件式が異なる。夫々を式（３－２）、（１３－２）、（１５－２）、（３５－２）、（３６－２）、（３７－２）、及び（３８－２）として以下に示す。

$end_n=sdata_{n-1}(8)$ …（３－２）

$plen_n \leqq 9$の時 $ext_n=0$ …（１２－２）

$plen_n > 9$の時 $ext_n=1$ …（１３－２）

$ext_n=1$ かつ $ocnt_n \neq 4$の時 $extptr_n = sum_n - (plen_n - 5)$ …（１５－２）

$end_{n-1}=0$ かつ $ext_{n-1}=1$の時 $buf_n(sptr_{n-1}: extptr_{n-1}-1) = sdata_{n-1}(0:8)$ …（３５－２）

$end_{n-1}=1$ かつ $ext_{n-1}=1$の時 $buf_n(sptr_{n-1}: extptr_{n-1}-1) = not(sdata_{n-1}(0:8))$ …（３６－２）

$end_{n-1}=0$ かつ $ext_{n-1}=1$の時 $buf_n(extptr_{n-1}:eptr_{n-1})= sdata_{n-1}(8), sdata_{n-1}(8),\cdots,sdata_{n-1}(8)$ …（３７－２）

$end_{n-1}=1$かつ$ext_{n-1}=1$の時 $buf_n(extptr_{n-1}:eptr_{n-1})= not(sdata_{n-1}(8), sdata_{n-1}(8),\cdots,sdata_{n-1}(8))$ …（３８－２）

【０２２２】

これらの演算式及び条件式を除けば、パッカ８０４の動作は図６で示したパッカ３０４の動作と同等であるため、説明を省略する。

【０２２３】

以上説明したように、連結シンボルデータに関しては１シンボルのみのデコード時を除

き、デコードテーブルに最後の変化点の次の画素まで格納し、パッカにてパック長分、前記最後の変化点の次の画素値を拡張することにした。これにより、連結シンボルデータのデコードテーブルのエントリー数の削減による、連結シンボルデータ用デコーダ及びテーブルの回路規模の削減が可能となる。また、シフト量及びパック長に関しては、連結符号語のヘッダ部分をデコードすることにより、シフト量及びパック長を算出するベース値を決定し、パック長を前記ベース値と、前記ヘッダ部分以外のビットとの総和で算出した。これにより、シフト量及びパック長用のデコードテーブルのエントリー数の削減による、シフト量及びパック長用デコーダ及びテーブルの回路規模の削減が可能になる。

【０２２４】

これにより、小さな回路規模で連結シンボル同時デコードが可能な、可変長復号化装置を提供することが可能となる。

【０２２５】

［第３の実施形態］

前記第１及び第２の実施形態における可変長符号化は１ビット／画素としてランレングス符号化を行っていた。しかしながら、多値の画素データにおいても（例えば８ビット／画素）、１ブロックを２色化し、入力画像が２色の代表色のどちらに属するかを識別する識別信号（２色の識別であるので１ビット／画素）と２色の代表色を符号化する構成においても適用できる。この場合は、図１１に示すように上記識別信号にランレングス符号化を用い、２色の代表色をブロックの先頭（または後端（EOBの直後））に配置すればよい。ランレングス符号化された識別信号は、前記第１及び第２の実施形態に示した方法にて復号化することで、高速、且つ、低コストに構成することができる。

【０２２６】

また、１ブロック２色以上の場合は、色の変化点をランレングス符号化し、ブロックの後端（ランレングス符号化のEOBの直後）に変化点毎の色を連結しておけばよい。例えば、図１２（ａ）に示すブロックデータを図１２（ｂ）に示すように横方向にジグザグに走査した場合、３箇所の色の境界（変化点）が発生する。図１３に上記図１２（ｂ）における符号化コードのパッキング例を示す。その境界までのラン長をランレングス符号化し（符号語数はEOBを含め４つ）、４色（境界＋１色）のデータをその後にパックする。但し、境界毎にEOB以降にパックする色は増えるので、ブロック内の色数に関らず（境界＋１色）分のパックが必要となる。

【０２２７】

［他の実施形態］

前記した各実施形態の機能を、パーソナルコンピュータ等のプロセッサを内蔵する情報処理装置や、マイコンを内蔵する装置が実行するコンピュータプログラムでもって実現させても構わない。また、通常、コンピュータプログラムは、ＣＤ－ＲＯＭ等のコンピュータ可読記憶媒体に格納されており、それをコンピュータが有する読取り装置（ＣＤ－ＲＯＭドライブ等）にセットし、システムにコピー若しくはインストールするとで実行可能になる。従って、かかるコンピュータ可読記憶媒体も本発明の範疇に入ることも明らかである。

【図面の簡単な説明】

【０２２８】

【図１】第１及び第２の実施形態における、単体シンボルの符号表である。

【図２】第１の実施形態における、デコードテーブルである。

【図３】第１の実施形態における、復号化装置の構成図である。

【図４】第１の実施形態における、連結シンボルデータ用デコーダ及びテーブルの動作を説明する図である。

【図５】第１の実施形態における、シフト量及びパック長用のデコーダ及びテーブルの動作を説明する図である。

【図６】第１の実施形態における、パッカの動作を説明する図である。

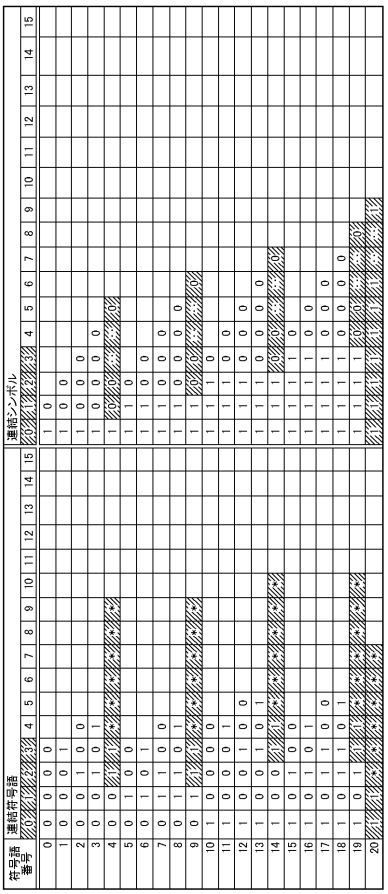
【図７Ａ】、

【図７Ｂ】第２の実施形態における、デコードテーブルである。
【図８】第２の実施形態における、復号化装置の構成図である。
【図９Ａ】、
【図９Ｂ】第２の実施形態における、連結シンボルデータ用デコーダ及びテーブルの動作を説明する図である。
【図１０Ａ】、
【図１０Ｂ】第２の実施形態における、シフト量及びパック長用のデコーダ及びテーブルの動作を説明する図である。
【図１１】第３の実施形態における、符号化コードのフォーマット例である。
【図１２】第３の実施形態における、多色データの入力例である。
【図１３】第３の実施形態における、多色データの符号化コードのフォーマット例である。
【図１４】実施形態における画像復号装置のブロック構成図である。
【図１５】１画素ブロックの符号化データのデータ構造を示す図である。

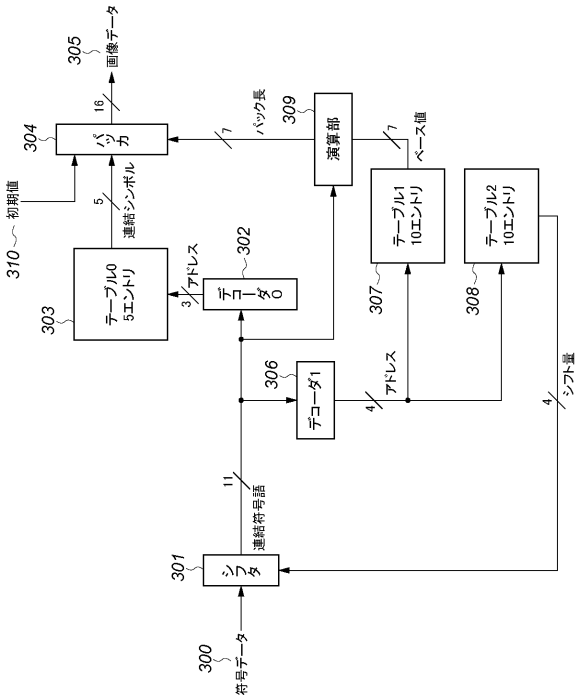
【図１】

単体符号語	単体シンボル(ラン長)
00	0
01	1
100	2
101	3
11000100-11111111	4-63
110000	EOB

【図２】



【図３】



【図４】

符号語番号	アドレス	連結符号語 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	連結シンボル 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0		0	0	0	0													1	0	0	0	0											
1		0	0	0	1													1	0	0	0	0											
2		0	0	1	0	0												1	0	0	0	0											
3		0	0	1	0	1												1	0	0	0	0											
4	0	0	0	1	1	0	0											1	1	0	0	0	0										
5		0	1	0	0	0	1											1	1	0	0	0	0										
6		0	1	0	0	1	0											1	1	0	0	0	0										
7		0	1	0	1	0	0											1	1	0	0	0	0										
8		0	1	0	1	0	1											1	1	0	0	0	0										
9	1	0	1	0	1	1	0	0										1	1	0	0	0	0										
10		1	0	0	0	0												1	1	1	0	0											
11		1	0	0	0	1												1	1	1	0	0											
12		1	0	0	1	0	0											1	1	1	0	0											
13		1	0	0	1	0	1											1	1	1	0	0											
14	2	1	0	0	1	1	0	0										1	1	1	1	0	0										
15		1	0	1	0	0	1											1	1	1	1	0	0										
16		1	0	1	0	1	0											1	1	1	1	0	0										
17		1	0	1	1	0	0											1	1	1	1	0	0										
18		1	0	1	1	0	1											1	1	1	1	0	0										
19	3	1	0	1	1	1	0	0										1	1	1	1	1	0										
20	4	1	0	1	1	1	1	0	0									1	1	1	1	1	0										

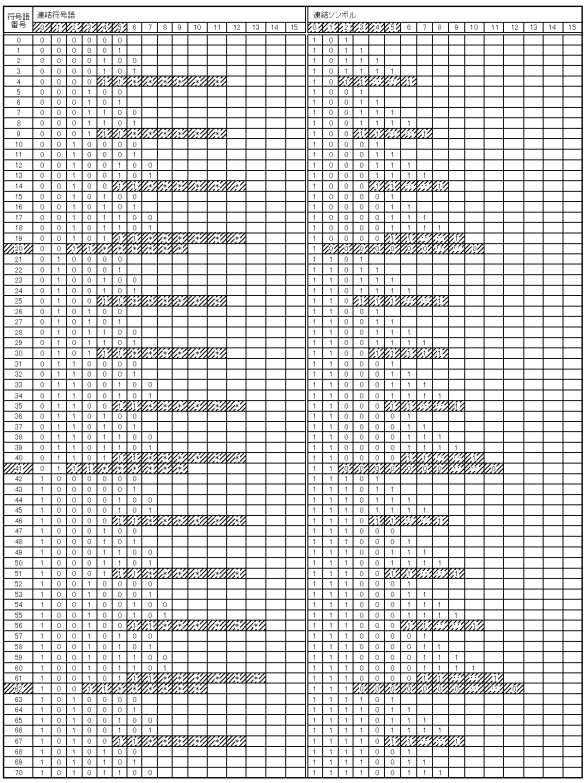
【図５】

符号語番号	アドレス	連結符号語 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	バック長	ベース値	10数	シフト量	連結シンボルデータ 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0		0	0	0	0													2	2	0	4	1	0														
1		0	0	1														3	2	1	4	1	1	0													
2		0	1	0														3	2	1	4	1	1	1	0												
3	0	0	1	1														4	2	2	4	1	1	1	1	0											
4		0	0	0	0													4	4	0	5	1	0	0	0	0											
5		0	0	0	1													5	4	1	5	1	1	0	0	0											
6		0	0	1	0													5	4	1	5	1	1	1	0	0											
7		0	0	1	1													6	4	2	5	1	1	1	1	0											
8	1	0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	EOB	64	0	6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
9		1	0	0	0													2+m	2	1	10	1	1														
10	3	1	0	0	1													3+m	2	1	10	1	1	1													
11		1	0	1	0													4	4	0	5	1	1	1	0	0											
12		1	0	1	1													5	4	1	5	1	1	1	1	0											
13	4	1	1	0	0													5	4	1	5	1	1	1	1	0											
14		1	1	0	1													6	4	2	5	1	1	1	1	0											
15		1	1	1	0	0												6	6	0	6	1	1	1	1	0											
16		1	1	1	0	1												7	6	1	6	1	1	1	1	0											
17	5	1	1	1	1	0												8	6	2	6	1	1	1	1	0											
18		1	1	1	1	1												EOB	64	0	9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
19	6	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	4+m	4	0	11	1	1	1													
20		0	0	0	0													5+m	4	1	11	1	1	1	1												
21	7	0	0	0	1													EOB	64	0	6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
22	8	0	0	1	0													1+m	1	0	8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
23	9	0	0	1	1																															

【図６】

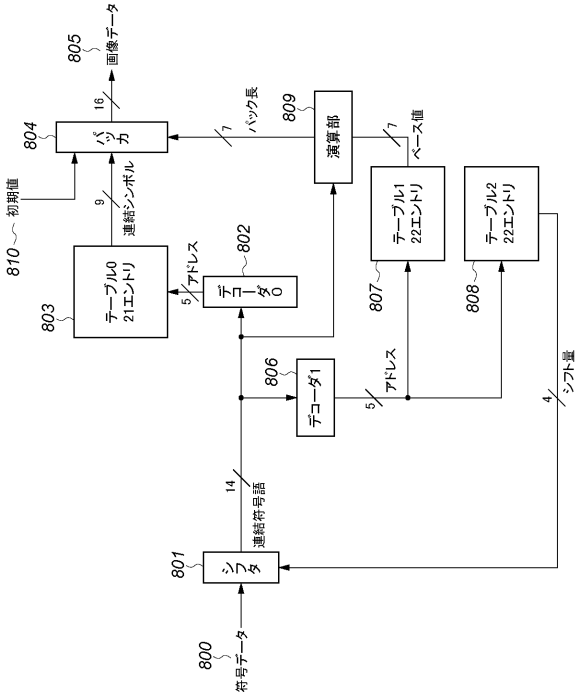
サイクル	sdata$_i$ 0	1	2	3	4	end$_i$	plen$_i$	sum$_i$	sptr$_i$	eptr$_i$	ext$_i$	exptr$_i$	pcnt$_i$	busy$_i$	rest$_i$	valid$_i$	ocnt$_i$	buf$_i$(0:15)	buf$_i$(16:31)	buf$_i$(32:47)	buf$_i$(48:63)	出力
0																						
1	1	0				0	0	0	0	0	0	0	0	0	0	0	0	'1----------	'1----------	'1----------	'1----------	-
2	1	0	0	0	0	0	2	2	0	1	0	2	0	0	0	0	0	'10---------	'10---------	'10---------	'10---------	-
3	1	0	0	0	0	0	5	7	2	6	0	7	0	0	0	0	0	'1010000----	'1010000----	'1010000----	'1010000----	-
4	1	1	1	1	0	0	7	14	7	13	1	12	0	0	0	0	0	'1010000110-	'1010000110-	'1010000110-	'1010000110-	-
5	1	1	1	1	0	0	17	31	14	30	1	19	1	0	1	0	1	'1010000111000001	'1000000000000000	'1000000000000000	buf$_i$(0:15)	
6	1	0	0	0	0	0	19	50	31	49	1	36	3	0	2	1	1	'1010000111000001	'1000000000000000	'10---------------	buf$_i$(16:31)	
7	1	0	0	0	0	0	64	114	50	63	1	55	4	1	2	1	2	'1010000111000001	'1000000000000000	'1100000000000000	buf$_i$(32:47)	
8	1	0	0	0	0	1	64	114	50	63	1	55	4	1	1	1	3	'1010000111000001	'1000000000000000	'1100000000000000	buf$_i$(48:63)	
	1	0	0	0	1	1	64	0	0	0	1	0	0	0	0	1	4	'1010000111000001	'1000000000000000	'1100000000000000	'0010000000000000	

【図７Ａ】

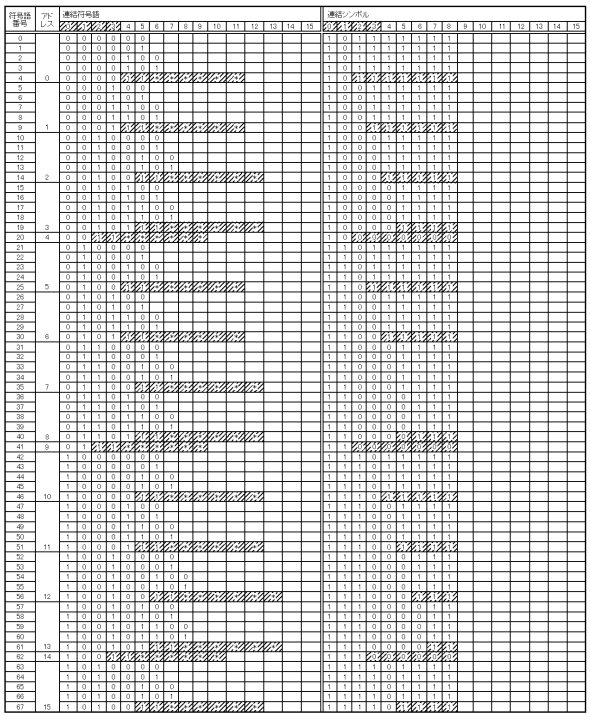


【図７Ｂ】

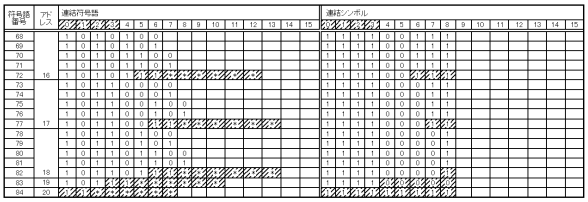
【図８】



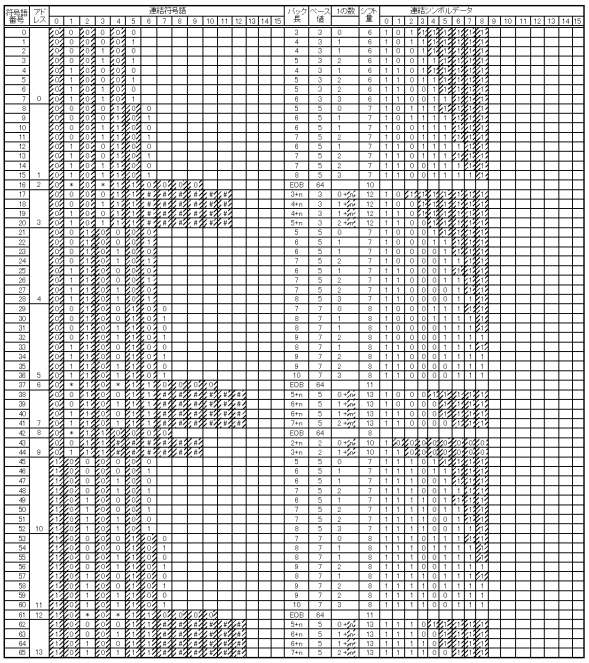
【図９Ａ】



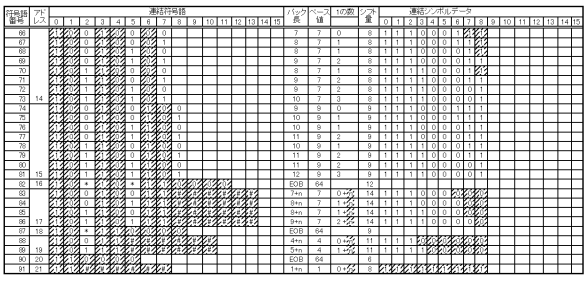
【図９Ｂ】



【図１０Ａ】

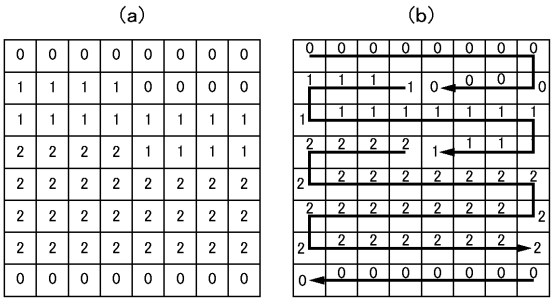


【図１０Ｂ】

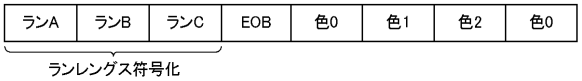


【図１１】

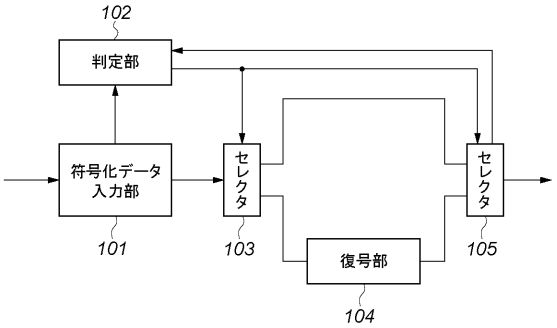
【図１２】



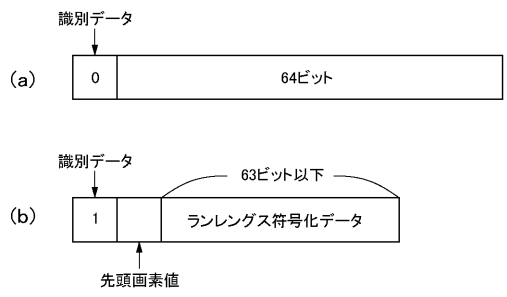
【図１３】



【図１４】



【図１５】



フロントページの続き

(72)発明者　原　裕司
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
(72)発明者　石川　尚
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

審査官　北村　智彦

(56)参考文献　特開平０８－３１７２２７（ＪＰ，Ａ）
特開２００１－２７４６９０（ＪＰ，Ａ）
国際公開第２００８／０８７７５０（WO，Ａ１）
特開２００７－１０４１９４（ＪＰ，Ａ）
特開２００３－１７４３６５（ＪＰ，Ａ）
特開２００２－２３７７５４（ＪＰ，Ａ）
特開平１０－０６５５４９（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｈ０３Ｍ　　３／００－１１／００
Ｈ０４Ｎ　　７／２６