(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4328358号
(P4328358)
(45)発行日　平成21年9月9日(2009.9.9)　(24)登録日　平成21年6月19日(2009.6.19)

(51)Int.Cl.　F I
H03M 7/40 (2006.01)　H03M 7/40

請求項の数 23　(全 48 頁)

(21)出願番号	特願2006-546744 (P2006-546744)
(86)(22)出願日	平成17年12月7日 (2005.12.7)
(86)国際出願番号	PCT/JP2005/022495
(87)国際公開番号	WO2006/062142
(87)国際公開日	平成18年6月15日 (2006.6.15)
審査請求日	平成19年3月29日 (2007.3.29)
(31)優先権主張番号	特願2004-354742 (P2004-354742)
(32)優先日	平成16年12月7日 (2004.12.7)
(33)優先権主張国	日本国(JP)

(73)特許権者　000004226
日本電信電話株式会社
東京都千代田区大手町二丁目３番１号
(74)代理人　100121706
弁理士　中尾　直樹
(74)代理人　100066153
弁理士　草野　卓
(74)代理人　100128705
弁理士　中村　幸雄
(72)発明者　原田　登
東京都千代田区大手町二丁目３番１号　日本電信電話株式会社内
(72)発明者　守谷　健弘
東京都千代田区大手町二丁目３番１号　日本電信電話株式会社内

最終頁に続く

(54)【発明の名称】情報圧縮符号化装置、その復号化装置、これらの方法、およびこれらのプログラムとその記録媒体

(57)【特許請求の範囲】

【請求項１】

一定ビット数の処理単位で表現された情報（以下、「単位処理情報」という。）の列に対応する、辞書部に登録された単位処理情報（以下、「辞書情報」という。）の列を探索して入力情報を圧縮符号化する情報圧縮符号化方法において、

前記入力情報から生成された単位処理情報列中の、入力情報の有効な桁に対応する有効なビットを、辞書情報の対応するビットと比較し、前記有効なビットと対応するビットとの全てが一致すれば、前記単位処理情報と辞書情報とが一致していると判定する一致判定ステップと

前記入力情報から生成された単位処理情報の列と一致していると判断できる辞書情報の列のうち最長の辞書情報の列を探索し、当該最長の辞書情報の列のインデックスを求める探索ステップと

前記インデックスまたは前記インデックスに対応する符号を出力する出力ステップと

を有する情報圧縮符号化方法。

【請求項２】

請求項１記載の情報圧縮符号化方法において、

前記インデックスが示す単位処理情報の列を含む単位処理情報の列を、辞書部に登録する辞書登録ステップ

も有する情報圧縮符号化方法。

【請求項３】

請求項２記載の情報圧縮符号化方法において、
前記辞書登録ステップでは、登録された単位処理情報の量が、所定の量よりも多くなった場合には、最も前に登録された単位処理情報から削除する
ことを特徴とする情報圧縮符号化方法。
【請求項４】
請求項１記載の情報圧縮符号化方法において、
入力情報がサンプル列であり、
サンプルごとに、サンプル中の有効な桁を構成するビット群を前記処理単位ごとに分割して前記単位処理情報を生成する単位処理情報生成ステップも有し、
前記単位処理情報生成ステップでは、分割によってサンプルのビットが余った場合もしくはサンプルのビットが処理単位に満たない場合には、余ったビットもしくは処理単位に満たないサンプルのビットにダミービットを追加して単位処理情報とし、追加したダミービット部分は有効なビットではないとする
ことを特徴とする情報圧縮符号化方法。
【請求項５】
請求項４記載の情報圧縮符号化方法において、
入力信号が浮動小数点形式であり、
入力信号を整数形式信号と差分信号とに分離する分離ステップと、
前記整数形式信号を圧縮符号化する整数形式符号化ステップ
も有し、
前記差分信号の列をサンプル列で構成される入力情報として、前記単位処理情報生成ステップ、前記一致判定ステップ、および前記探索ステップを行い、
前記出力ステップでは、前記インデックスに加えて、前記整数形式符号化ステップで圧縮符号化された符号も出力する
ことを特徴とする情報圧縮符号化方法。
【請求項６】
請求項５記載の情報圧縮符号化方法において、
前記分離ステップで分離された前記差分信号が、前記整数形式信号が０の場合の差分信号であるかを判定するステップと、
前記判定で整数形式信号が０の場合の差分信号であると判定されると、当該差分信号により、第１の差分信号列を生成するステップと、
前記判定で整数形式信号が０でない場合の差分信号であると判定されると、当該差分信号により、第２の差分信号列を生成するステップ
も有し、
前記第１の差分信号列と第２の差分信号列とを連結したものを前記サンプル列で構成される入力情報とする
ことを特徴とする情報圧縮符号化方法。
【請求項７】
請求項４記載の情報圧縮符号化方法において、
所定の複数の入力信号により構成される分割区間ごとの入力信号列を、共通の乗数、その乗数で各入力信号を除算処理した商信号の列、その残りの差分信号の列とに分解する分解ステップと、
前記商信号の列を圧縮符号化して出力する商信号符号化ステップと、
前記乗数を符号化して出力する乗数符号化ステップ
も有し、
前記差分信号の列をサンプル列で構成される入力情報として、前記単位処理情報生成ステップ、前記一致判定ステップ、前記探索ステップ、および前記出力ステップを行う
ことを特徴とする情報圧縮符号化方法。
【請求項８】
請求項７記載の情報圧縮符号化方法において、

前記分解ステップで得られた前記差分信号の列を構成する差分信号が、前記商信号が０の場合の差分信号であるかを判定するステップと、
前記判定で商信号が０の場合の差分信号であると判定されると、当該差分信号により、第１の差分信号列を生成するステップと、
前記判定で商信号が０でない場合の差分信号であると判定されると、当該差分信号により、第２の差分信号列を生成するステップ
も有し、
前記第１の差分信号列と第２の差分信号列とを連結したものを前記サンプル列で構成される入力情報とする
ことを特徴とする情報圧縮符号化方法。

【請求項９】
請求項４記載の情報圧縮符号化方法において、
所定の複数の浮動小数点入力信号により構成される分割区間ごとの浮動小数点入力信号列を、共通の乗数、その乗数で各浮動小数点入力信号を除算処理した信号の列とに分解する分解ステップと、
前記除算処理した信号を、整数形式信号に変換する整数化ステップと、
前記整数形式信号に前記乗数を乗算した浮動小数点信号を求める乗算ステップと、
前記乗算ステップで得られた浮動小数点信号と前記浮動小数点入力信号との差分信号を求める差分信号算出ステップと、
前記整数形式信号の列を圧縮符号化して出力する整数形式信号符号化ステップと、
前記乗数を符号化して出力する乗数符号化ステップ
も有し、
前記差分信号の仮数部の列をサンプル列で構成される入力情報として、前記単位処理情報生成ステップ、前記一致判定ステップ、前記探索ステップ、および前記出力ステップを行う
ことを特徴とする情報圧縮符号化方法。

【請求項１０】
請求項４記載の情報圧縮符号化方法において、
所定の複数の浮動小数点入力信号により構成される分割区間ごとの浮動小数点入力信号列を、共通の乗数、その乗数で各浮動小数点入力信号を除算処理した信号の列とに分解する分解ステップと、
前記除算処理した信号を、整数形式信号に変換する整数化ステップと、
前記整数形式信号に前記乗数を乗算した浮動小数点信号を求める乗算ステップと、
前記整数形式信号の列を圧縮符号化して出力する整数形式信号符号化ステップと、
前記乗数を符号化して出力する乗数符号化ステップ
も有し、
前記乗数が１であり、かつ前記整数形式信号が０でない場合は、
前記整数形式信号の有効桁数ｎにより定まる前記浮動小数点入力信号の仮数部の下位（前記浮動小数点入力信号の仮数部の桁数－ｎ）ビットを有効なビットとして含むビット群の列を、サンプル列で構成される入力情報として、前記単位処理情報生成ステップ、前記一致判定ステップ、前記探索ステップ、および前記出力ステップを行い、
前記乗数が１ではなく、かつ前記整数形式信号が０でない場合は、
前記乗算ステップで得られた乗算済の浮動小数点信号と前記浮動小数点入力信号との差分信号を求める差分信号算出ステップも有し、
前記差分信号算出ステップで得た差分信号の仮数部の列をサンプル列で構成される入力信号として、前記単位処理情報生成ステップ、前記一致判定ステップ、前記探索ステップ、および前記出力ステップを行い、
前記整数形式信号が０の場合は、
前記浮動小数点入力信号の列を、サンプル列で構成される入力信号として、前記単位処理情報生成ステップ、前記一致判定ステップ、前記探索ステップ、および前記出力ステッ

プを行う
ことを特徴とする情報圧縮符号化方法。
【請求項１１】
一定ビット数の処理単位で表現された情報（以下、「単位処理情報」という。）の列が登録されている辞書部のインデックスの、いずれか１つのインデックスを含む入力符号を復号化して、復号情報を得る情報復号化方法において、
前記入力符号に含まれる情報から、前記単位処理情報ごとの有効なビットを示す情報（以下、「有効桁情報」という。）を求める有効桁情報取得ステップと、
前記入力符号に含まれるインデックスから、辞書部に登録されている単位処理情報の列を取得する単位処理情報列取得ステップと、
前記有効桁情報を用いて、前記取得された単位処理情報列中の有効部分を特定して復号情報を得るマスク処理ステップと
を有する情報復号化方法。
【請求項１２】
一定ビット数の処理単位で表現された情報（以下、「単位処理情報」という。）の列が登録されている辞書部のインデックスの、いずれか１つのインデックスを含む入力符号を復号化して、復号情報を得る情報復号化方法において、
前記入力符号に含まれる情報から、前記単位処理情報ごとの有効なビットを示す情報（以下、「有効桁情報」という。）を求める有効桁情報取得ステップと、
前記入力符号に含まれるインデックスから、辞書部に登録されている単位処理情報の列を取得する単位処理情報列取得ステップと、
前記有効桁情報を用いて、前記取得された１つまたは複数の単位処理情報列に含まれる１つまたは複数の単位処理情報の有効なビットを結合したものを復号信号とする結合ステップと
を有する情報復号化方法。
【請求項１３】
請求項１２記載の情報復号化方法において、
入力符号を整数符号と差分符号とに分離する分離ステップと、
前記整数符号を復号して整数形式信号を生成する整数形式信号復号ステップと、
前記整数形式信号を浮動小数点形式信号に変換する浮動小数点化ステップと、
前記浮動小数点形式信号の仮数部と前記結合ステップの出力とを合成したものに、仮数部を置き換えた前記浮動小数点形式信号を復号信号とする合成ステップ
も有し、
前記有効桁情報取得ステップおよび前記単位処理情報列取得ステップは、前記差分符号に含まれる情報またはインデックスを処理の対象とする
ことを特徴とする情報復号化方法。
【請求項１４】
請求項１２記載の情報復号化方法において、
入力符号を商符号と乗数符号と差分符号とに分離する分離ステップと、
前記商符号を復号化して商信号を求める商復号化ステップと、
前記乗数符号を復号化して乗数を求める乗数復号化ステップと、
前記乗数を前記商信号に乗算する乗算ステップと、
前記乗算ステップの出力と前記結合ステップの出力とを加算したものを復号信号とする加算ステップ
も有し、
前記有効桁情報取得ステップおよび前記単位処理情報列取得ステップは、前記差分符号に含まれる情報またはインデックスを処理の対象とする
ことを特徴とする情報復号化方法。
【請求項１５】
請求項１２記載の情報復号化方法において

入力符号を整数符号と乗数符号と差分符号とに分離する分離部ステップと、
前記整数符号を復号化して整数信号を求める整数信号復号化ステップと、
前記乗数符号を復号化して乗数を求める乗数復号化ステップと、
前記乗数が１であるか否かを判定する乗数判定ステップ
も有し、
前記単位処理情報列取得ステップは、差分符号に含まれるインデックスを処理の対象とするものであり、
前記乗数が１であり、かつ前記整数信号が０でない場合は、
前記有効桁情報取得ステップは、前記整数信号の有効桁数ｎを有効桁情報として求めるものであり、
前記結合ステップで得られた復号信号を前記有効桁情報に基づく仮数部の下位（出力信号の仮数部の桁数－ｎ）ビット部分とした浮動小数点形式信号を出力するステップも有し、
前記乗数が１ではなく、かつ前記整数信号が０でない場合は、
前記整数信号に前記乗数を乗算する乗算ステップと、
前記乗算ステップの出力と前記結合ステップで得られた復号信号とを加算したものを出力信号とする加算ステップも有し、
前記整数信号が０の場合は、
前記結合ステップで得られた復号信号を仮数部とした浮動小数点形式信号を出力信号とするステップも有する
ことを特徴とする情報復号化方法。

【請求項１６】
一定ビット数の処理単位で表現された情報（以下、「単位処理情報」という。）の列に対応する、辞書部に登録された単位処理情報（以下、「辞書情報」という。）の列を探索して入力情報を圧縮符号化する情報圧縮符号化装置において、
前記入力情報から生成された単位処理情報列中の、入力情報の有効な桁に対応する有効なビットを、辞書情報の対応するビットと比較し、前記有効なビットと対応するビットとの全てが一致すれば、前記単位処理情報と辞書情報とが一致していると判定する一致判定部と
前記入力情報から生成された単位処理情報の列と一致していると判断できる辞書情報の列のうち最長の辞書情報の列を探索し、当該最長の辞書情報の列のインデックスを求める探索部と
前記インデックスを出力する出力部と
を備える情報圧縮符号化装置。

【請求項１７】
請求項１６記載の情報圧縮符号化装置において、
入力情報がサンプル列であり、
サンプルごとに、サンプル中の有効な桁を構成するビット群を前記処理単位ごとに分割して前記単位処理情報を生成する単位処理情報生成部も備え、
前記単位処理情報生成部では、分割によってサンプルのビットが余った場合もしくはサンプルのビットが処理単位に満たない場合には、余ったビットもしくは処理単位に満たないサンプルのビットにダミービットを追加して単位処理情報とし、追加したダミービット部分は有効なビットではないとする
ことを特徴とする情報圧縮符号化装置。

【請求項１８】
一定ビット数の処理単位で表現された情報（以下、「単位処理情報」という。）の列が登録されている辞書部のインデックスの、いずれか１つのインデックスを含む入力符号を復号化して、復号情報を得る情報復号化装置において、
前記入力符号に含まれる情報から、前記単位処理情報ごとの有効なビットを示す情報（以下、「有効桁情報」という。）を求める有効桁情報取得部と、

前記入力符号に含まれるインデックスから、辞書部に登録されている単位処理情報の列を取得する単位処理情報列取得部と、
前記有効桁情報を用いて、前記取得された１つまたは複数の単位処理情報列に含まれる１つまたは複数の単位処理情報の有効なビットを結合したものを復号信号とする結合部と
を備える情報復号化装置。
【請求項１９】
請求項１８記載の情報復号化装置において、
入力符号を整数符号と差分符号とに分離する分離部と、
前記整数符号を復号して整数形式信号を生成する整数形式信号復号部と、
前記整数形式信号を浮動小数点形式信号に変換する浮動小数点化部と、
前記浮動小数点形式信号の仮数部と前記結合ステップの出力とを合成したものに、仮数部を置き換えた前記浮動小数点形式信号を復号信号とする合成部
も備え、
前記有効桁情報取得部および前記単位処理情報列取得部は、前記差分符号に含まれる情報またはインデックスを処理の対象とする
ことを特徴とする情報復号化装置。
【請求項２０】
請求項１８記載の情報復号化装置において、
入力符号を商符号と乗数符号と差分符号とに分離する分離部と、
前記商符号を復号化して商信号を求める商復号化部と、
前記乗数符号を復号化して乗数を求める乗数復号化部と、
前記乗数を前記商信号に乗算する乗算部と、
前記乗算部の出力と前記結合部の出力を加算したものを復号信号とする加算部
も有し、
前記有効桁情報取得部および前記単位処理情報列取得部は、前記差分符号に含まれる情報またはインデックスを処理の対象とする
ことを特徴とする情報復号化装置。
【請求項２１】
請求項１６記載の情報圧縮符号化装置において、
複数のハッシュテーブルと、
ｋ文字の長さの単位処理情報が辞書に登録されている場合に、不一致文字を追加する際に登録されている辞書のインデックスと、不一致文字をハッシュキーとしてハッシュテーブルに登録する切替部と
を有し、
前記探索部は、１つの辞書登録に対して、最長一致文字の次の１文字の有効桁情報を変化させ、有効桁情報に応じてハッシュテーブルを選択して登録を行っておき、探索の際に有効桁情報に対応するハッシュテーブルを用いて探索を行う
ことを特徴とする情報圧縮符号化装置。
【請求項２２】
請求項１、１１、１２のいずれかに記載した方法の各過程をコンピュータに実行させるためのプログラム。
【請求項２３】
請求項１、１１、１２のいずれかに記載した方法の各過程をコンピュータに実行させるプログラムを記録したコンピュータ読取可能な記録媒体。
【発明の詳細な説明】
【技術分野】
【０００１】
この発明は、ＬＺ等の辞書を用いて情報を圧縮符号化する装置、その復号化装置、これらの方法およびプログラムとその記録媒体に関する。
【背景技術】

【０００２】
近年、文字列データ、音響信号データや画像情報データを、通信路により伝送する場合や情報記録媒体に記録する場合に、情報圧縮符号化を用いてデータの情報量を減らすことが行われている。
一定数の単位のビットを１文字と見て、文字列の繰り返しをモデルとして情報圧縮を行う符号化方法がある（例えば非特許文献１参照）。このような符号化方法にはＬＺ７７、ＬＺ８８、ＬＺＷ等の方法がある（例えば非特許文献１、非特許文献２参照）。
文字列の繰り返しをモデルとする情報圧縮符号化方法は、テキスト文書等の圧縮符号化のみならず、モデム信号、画像信号の符号化方法としても利用されている。
【０００３】
図１ＡにＬＺ７７法による文字列の情報圧縮符号化装置の例を、図１Ｂにその復号化装置の例を示す。ＬＺ７７法の符号化装置では、辞書登録部１２は、入力端子１１からの入力文字列を、辞書部１３に一定量保持しておく。一致探索部１４は、新たに入力された文字列と辞書部１３内の過去の文字列とを比較する。符号生成部１５は、一定の文字数以上で、最も長く一致する文字列（最長一致文字列）の辞書部（バッファ）１３上の位置（インデックス）と、その一致した長さと、入力文字列中の最長一致文字の次の１文字（不一致文字）とを、符号として出力端子１６へ出力する。また、符号生成部１５は、最長一致文字列の長さが一定の文字数未満であった場合には、不一致をあらわす符号、例えばバッファ上の位置（インデックス）０及び一致した長さ０の符号とその一致長が一定文字数未満の入力文字列の先頭の１文字とを出力する。
【０００４】
辞書部１３は、例えば図１Ｃに示すように、その一端に先読みバッファ１７が接続され、全体としてスライドバッファが構成されている。先読みバッファ１７の一端、図では右端から入力文字列が順次格納され、先読みバッファ１７が満杯になると、先読みバッファ１７の左端から文字列に対する一致探索処理が開始される。符号が出力されると、最長一致文字列の長さの文字又は１文字が辞書部１３の左端から廃棄され、辞書部１３の右端に最後に符号化された文字列又は１文字が格納される。また、その分先読みバッファ１７の右端に入力文字列が格納される。
【０００５】
ＬＺ７７法の復号化装置では、符号解析部２２は、入力端子２１からの入力符号が不一致をあらわす符号であるか否かの判別を行う。不一致をあらわす符号でなければ、符号解析部２２は、入力文字列を位置情報（インデックス）と、一致連続長と、不一致の１文字（不一致文字）とに分離する。文字列取得部２３は、位置情報と一致連続長に基づき辞書部２４から対応する文字列を取り出す。また、文字列合成部２５は、辞書部２４から取り出された文字列と不一致文字を合成し、復号文字列として出力端子２６に出力する。さらに、辞書登録部２７は、復号文字列を辞書部２４に格納する。辞書部２４は、スライドバッファであり、文字列が図１Ａ中の辞書部１３と相対的に同一状態で保持される。
【０００６】
符号解析部２２は、不一致をあらわす符号が入力された場合には、これに続く不一致文字をそのまま出力すると共に辞書部２４に格納する。
ＬＺ７７を改良した符号化方法にＬＺＳＳ、ＺＩＰ等がある。ＺＩＰ等ではＬＺ７７で得られた符号出力を、さらにエントロピー符号化の一種であるハフマン符号化で圧縮する。
【０００７】
図２Ａに代表的なＬＺ７８方法を用いた文字列の情報圧縮符号化装置の例を、図２Ｂにその復号化装置の例をそれぞれ示す。ＬＺ７８符号化では、辞書部３１が空の状態から処理を開始する。一致探索部３２は、入力端子１１から新たに入力された文字列と、辞書部３１に登録された文字列とを比較する。符号生成部３３は、最も長く一致する最長一致文字列に対応した辞書部３１のインデックスと、次の一致しなかった入力１文字に対応するインデックスとを組にし、符号として出力端子１６に出力する。また、辞書登録部３４は

、符号を出力する毎に、前記最長一致文字列に次の１文字を加えた文字列を、辞書部３１に新たに登録していく。辞書部３１は、スライドバッファではなく、固定である。図２Ａの符号化装置は、入力された文字が辞書部３１になかった場合には、不一致をあらわす特殊なインデックスと、１文字をそのまま符号として出力する。

【０００８】

図２Ｂに示すＬＺ７８復号化装置では、辞書部９４１は符号化に用いた辞書部３１の初期状態と同じ初期状態である。つまり辞書部９４１は、空の状態から処理を開始する。符号解析部９４２は、入力符号を最長一致文字列のインデックスと続く１文字の不一致文字インデックスとに分離する。文字列取得部９４３は、インデックスを用いて辞書部９４１から一致文字列と不一致文字を取り出す。文字列合成部２５は、取り出された一致文字列と不一致文字とを合成することで文字列を復号し、出力端子２６に出力する。この際、辞書登録部９４４は、最長一致文字列に不一致文字の１文字を加えた文字列を、符号化装置が行ったのと同じ方法で、辞書部９４１に新たに登録する。このようにして復号化装置の辞書部９４１の内部状態と、符号化装置の辞書部３１の内部状態とを同一に保つ。

【０００９】

ＬＺ７８符号化では、辞書部３１，９４１は木構造で構成されている。辞書内に入力文字列と等しい要素が登録されているかどうかを探索する際に、ハッシュ関数で文字列を圧縮するなどして探索を高速化する方法も知られている。

ＬＺＷの符号化装置を図３Ａに、その復号化装置を図３Ｂにそれぞれ示す。ＬＺＷの符号化装置の辞書部３５には、限られた数の初期文字（データ）が予め登録されている。一致探索部３６は、入力された文字列と辞書部３５に登録されている文字列とを比較する。そして、一致探索部３６は、最も長く一致する最長一致文字列と、その文字列に対応した辞書部３５内のインデックスを、符号生成部３７に出力する。符号生成部３７は、辞書部３５内のインデックスを符号として出力端子１６から出力する。また、符号生成部３７は、最長一致文字列を辞書登録部３８に出力する。その文字列の先頭文字C_Fが、直前バッファ３８ａ内に一時的に保持されている直前の最長一致文字列の最後に付けられて、辞書部３５に登録される。つまり前回符号化した文字列と、今回符号化した文字列の先頭文字C_Fとが１つの連続文字列として、辞書部３５に登録されることになる。

【００１０】

次に図３Ｂの復号化装置について説明する。符号解析部９４６は、端子２１からの入力符号から、最長一致文字列のインデックスを順次取り出す。文字列取得部９４７は、取り出されたインデックスと対応した文字列を、辞書部９４５から読み出す。文字列合成部９４８は、辞書部９４５から読み出された文字列を、復号文字列として出力端子２６へ出力する。同時に、辞書登録部９４９は、今回復号された文字列の先頭文字C_Fを、直前バッファ９４９ａ内の前回復号された文字列の後に付け、辞書部９４５に登録する。

【００１１】

一方、音響信号データ（ディジタルサンプル値列）向けの歪を許す圧縮符号化方法としてＭＰ３、ＡＡＣ、TwinＶＱ等がある。画像情報データ（サンプル値）向けの圧縮符号化方法としてＪＰＥＧ等がある。歪を許さない可逆な符号化（ロスレス符号化）技術もある（例えば非特許文献３参照）。更に編集加工が容易な浮動小数点形式のデータの可逆圧縮も重要である。特に浮動小数点形式オーディオ信号を、整数列と、その残り（差分）部分の仮数部中の非ゼロとなり得るビットのみに分け、それぞれ符号化して圧縮効率を向上させたものがある（例えば非特許文献４参照）。

【００１２】

元のサンプル列の各サンプルｓ０（ｉ）に、ゲインとして実数Ｇを共通に掛け合わせて得られたサンプルｓ１（ｉ）は、ｓ１（ｉ）＝ｓ０（ｉ）×Ｇとなる。この実数Ｇを掛け合わせて得られたサンプルｓ１（ｉ）を、２進表現やＩＥＥＥ－７５４形式の２進浮動小数点表現で表現し、これらディジタルサンプル列を符号化することがよくある。ＩＥＥＥ－７５４として標準化されている浮動小数点形式は、図４に示すように３２ビットであり、上位ビットから極性１ビット、指数部８ビット、仮数部２３ビットで構成される。極性

をＳ、指数部８ビットで表す値を１０進数でＥ、仮数部の２進数をＭとすると、この浮動小数点形式の数値を絶対値表現２進数で表わすと式（１）となる。

【００１３】

【数１】

$$(-1)^S \times 1.M \times 2^{E-E_0} \qquad (1)$$

ＩＥＥＥ－７５４によれば、$E_0 = 2^7 - 1 = 127$と決められており、式（１）中のＥ－E_0は

$$-127 \leqq E - E_0 \leqq 128$$

の範囲の任意の値を取ることができる。ただし、$E - E_0 = 127$の場合は全て“０”、$E - E_0 = 128$の場合は全て“１”と定義されている。$E - E_0 = n$は式（１）で表される値の整数部分の桁数（ビット数）から１を減算した値、即ち、最上位の“１”より下位ビット数を表わしている。

【非特許文献１】Nelson, Gailly（荻原、山口訳）「データ圧縮ハンドブック改定第２版」第７章～第９章

【非特許文献２】D. Salomon“Data Compression”, pp.101-162(Chapter3)

【非特許文献３】Hans, M.and Schafer, R.W.：Lossless Compression of Digital Audio, IEEE Signal Processing Magazine,Vol.18,No.4,pp.21-32(2001)

【非特許文献４】Dai Yang, and Takehiro Moriya：Lossless Compression for Audio Data in the IEEE Floating-Point Format, AES Convention Paper 5987, AES 115th Convention, New York, NY, USA, 2003 OCTOBER 10-13

【発明の開示】

【発明が解決しようとする課題】

【００１４】

文字列の繰り返しをモデルとする情報圧縮符号化方法では、基本としているモデルの特性から、同様の文字の組み合わせが繰り返し出現するような場合に圧縮率が向上する。

オーディオ信号のディジタルサンプル値、画像信号のディジタルサンプル値、心電波ディジタルサンプル値、地震波ディジタルサンプル値などの計測信号のディジタルサンプル値、線形予測分析の残差信号のディジタル値、その他のディジタル数値などのディジタル情報についてもＬＺなどの辞書を用いて圧縮符号化することが考えられる。しかし、現実には、同様の組み合せ（ディジタル値）の出現する場合が比較的少なく、所望の圧縮率を得ることができない場合が多かった。

【００１５】

この発明はこの問題を解決する情報圧縮符号化装置、その復号化装置、これらの方法、これらのプログラム及びその記録媒体を提供することを目的とする。

【課題を解決するための手段】

【００１６】

この発明によれば、一定ビット数の処理単位で表現された情報を、ＬＺ等の辞書を用いる情報圧縮符号化方法において、有効なビット桁数（以下有効桁長という）が既知の入力情報と辞書に登録されている情報と比較し、入力情報中の前記有効桁長のビットが、全て一致すれば、その登録情報のインデックスを入力情報の符号の少くとも一部として出力する。

【発明の効果】

【００１７】

２進表現された情報、“０”と“１”の２値の配列で表現された情報の情報集合をよくみてみると、各情報中にはその２値の一方に片寄りがある場合が比較的多いことに気付いた。従来は入力情報が辞書の登録情報とが完全に一致しているものを探索していた。しかし、この発明では片寄りのある情報中の重要な部分つまり入力情報を構成するビットのうち、少なくとも有効桁に相当する部分のビットが辞書中の登録情報と一致するものを探し

、一致したそのインデックスが符号出力とされる。したがって、探索により同一インデックスが見い出される場合が多く、つまり同一インデックスの符号が出力される場合が多くなり、それだけ従来より圧縮率が高くなる。

【発明を実施するための最良の形態】

【００１８】

以下にこの発明の実施例を図面を参照して説明するが、図面中の対応する部分は同一参照番号を付けて重複説明を省略する。

［実施例１］

符号化構成

入力情報（データ）の有効ビット桁数が知られている情報を符号化する場合について説明する。入力情報としては各種のものが考えられ、入力情報は、一般には文字ではない。しかし、本発明は、図１～図３に示した従来技術と同様に辞書を用いて符号化するものである。そこで、理解し易さの観点から、以下の説明では、符号化の対象である入力情報のビット列を複数ビットごと（例えば８ビットごと）にまとめたものを文字と表現する。また、この実施例では有効桁情報としてマスクデータが用いられる場合を例とする。例えば、入力文字（処理単位情報）Ｃ１（ｉ）が、０１０１０１０１であり、そのマスクデータＭ１（ｉ）は１１１１０００００である。マスクデータＭ１中の“１”は文字ビット列中の対応桁が有効であり、“０”は対応桁が有効でない又は無視してもよいことを表わす。つまりこの例では文字Ｃ１（ｉ）中の上位４ビット０１０１は有効であるが、下位４ビット０１０１は有効でない又は無視してもよいことを表わしている。

【００１９】

実施例１の機能構成を図５に示す。端子４１$_{C}$からは、入力文字Ｃ１（ｉ）を先頭とする入力文字列Ｃ１（ｉ＋０）～Ｃ１（ｉ＋ｋ）が１文字ずつ逐次入力される。また、端子４１$_{M}$からは、入力文字列Ｃ１（ｉ＋０）～Ｃ１（ｉ＋ｋ）に対応したマスクデータ列（有効桁情報）Ｍ１（ｉ＋０）～Ｍ１（ｉ＋ｋ）が１文字分ずつ逐次入力される。ここでｋは０から1ずつ増加する正整数である。

辞書部４３には、各インデックスに対して文字列が登録されている。この例では各インデックスｊに対して、Ｋ$_{j}$個の文字からなる文字列Ｄ（ｊ）（０）～Ｄ（ｊ）（Ｋ$_{j}$－１）が登録されている。辞書のインデックスｊに対応する格納領域に登録された文字列を構成する各文字は、ｋを０≦ｋ≦Ｋ$_{j}$－１とするとき、Ｄ（ｊ）（ｋ）と表現する。文字列を、Ｄ（ｊ）（ｋ）［ｋ：０≦ｋ≦Ｋ$_{j}$－１］またはＤ（ｊ）（ｋ）［ｋ：０≦ｋ＜Ｋ$_{j}$］と表現する。

【００２０】

一致探索部４２は、有効桁情報列Ｍ１（ｉ＋０）～Ｍ１（ｉ＋Ｋ$_{j}$－１）を考慮した上で、入力文字列Ｃ１（ｉ＋０）～Ｃ１（ｉ＋Ｋ$_{j}$－１）と、辞書部４３のｊ番目のインデックスに登録されている文字列Ｄ（ｊ）（０）～Ｄ（ｊ）（Ｋ$_{j}$－１）とを、先頭の文字から比較する。具体的には、一致探索部４２は、ｋ番目の文字Ｃ１（ｋ）とＤ（ｊ）（ｋ）の有効桁情報Ｍ１（ｉ＋ｋ）の構成ビットのうち１となっているビットに対応するビットが一致していれば、一致すると判断する。一致すると判断した場合は、ｋを１ずつ増加させながら有効桁部分が入力文字列と完全に一致する辞書部に登録されている文字列のうち、最長のものの辞書インデックスｊを探索する。そして、探索の結果、得られたインデックスｊが出力端子４４に圧縮符号の少なくとも一部として出力される。

【００２１】

以下に文字とそのマスクデータとの論理積をマスク処理として少なくとも有効桁部分が一致する辞書登録文字を探索する場合の実施例の詳細を示す。

入力文字Ｃ１（ｉ＋ｋ）と有効桁情報Ｍ１（ｉ＋ｋ）が、ｋを０から１ずつ増加させながらマスク計算部４２ａに逐次入力される。マスク計算部４２ａは、Ｃ１（ｉ＋ｋ）とＭ１（ｉ＋ｋ）とのビットごとの論理積Ｃ１（ｉ＋ｋ）＆Ｍ１（ｉ＋ｋ）を計算する。

また、有効桁情報Ｍ１（ｉ＋ｋ）は、マスク計算部４８にも入力される。マスク計算部４８は、辞書部４３のｊ番目のインデックスに登録された文字列Ｄ（ｊ）（ｋ）［ｋ：０

≦ｋ＜Ｋ$_{j}$］中の文字Ｄ（ｊ）（ｋ）とＭ１（ｉ＋ｋ）とのビットごとの論理積Ｄ（ｊ）（ｋ）＆Ｍ１（ｉ＋ｋ）を計算する。

【００２２】

以下では、文字Ｃ１（ｉ）をその有効桁情報Ｍ１（ｉ）でマスク処理した結果Ｃ１（ｉ）＆Ｍ１（ｉ）をマスク文字という。また、同様に、文字Ｄ（ｊ）（ｉ）を対応する有効桁情報Ｍ１（ｉ）でマスク処理した結果Ｄ（ｊ）（ｉ）＆Ｍ１（ｉ）もマスク文字という。

情報一致判定部４２ｂは、マスク計算部４２ａからの各マスク文字と、マスク計算部４８よりの各マスク文字とを、ｋ＝０からｋを１ずつ増加させながら逐次比較する。そして、Ｃ１（ｉ＋ｋ）＆Ｍ１（ｉ＋ｋ）とＤ（ｊ）（ｋ）＆Ｍ１（ｉ＋ｋ）が、ｋ＝０～Ｋ$_{j}$－１の全てで一致した場合に、情報一致判定部４２ｂは、辞書インデックスｊに登録された文字列と入力文字列が一致したと判定する。

【００２３】

これは、Ｃ１（ｉ＋ｋ）とＤ（ｊ）（ｋ）の一致判定を行う際に、有効桁情報Ｍ１（ｉ＋ｋ）の構成ビットのうち、１となって有効であるとされるビットが少なくとも一致していればこれら２つの文字が一致したと判定することに相当する。

たとえば、Ｃ１（ｉ＋ｋ）が、０１０１０１０１であり、そのマスクデータＭ１（ｉ＋ｋ）が１１１１０００、Ｄ（ｊ）（ｋ）が０１０１１０１０であるとき、Ｃ１（ｉ＋ｋ）＆Ｍ１（ｉ＋ｋ）は０１０１０００となり、Ｄ（ｊ）（ｋ）＆Ｍ１（ｉ＋ｋ）も０１０１０００となる。これによってマスク文字Ｃ１（ｉ＋ｋ）＆Ｍ１（ｉ＋ｋ）とマスク文字Ｄ（ｊ）（ｋ）＆Ｍ１（ｉ＋ｋ）は一致したと判定される。すなわち、Ｃ１（ｉ＋ｋ）とＤ（ｊ）（ｋ）は少なくとも有効桁に相当する部分のビットが一致したことになる。

【００２４】

ここで辞書部４３には過去に入力された文字列を符号化して出力された文字列が登録されている。

たとえば、ｘ文字数だけ過去に入力されたＫ文字の文字列をＣ１（ｉ－ｘ＋ｋ）［ｋ：０≦ｋ≦Ｋ－１］とする。マスク計算部４２ａから出力された文字列はＣ１（ｉ－ｘ＋ｋ）＆Ｍ１（ｉ－ｘ＋ｋ）［ｋ：０≦ｋ≦Ｋ－１］とあらわされる。この場合、辞書部４３のインデックスｊにｘ$_{j}$文字数だけ過去の文字列が登録されているとすれば、登録された文字列を構成する各文字は次式であらわされる。

【００２５】

Ｄ（ｊ）（ｋ）＝Ｃ１（ｉ－ｘ$_{j}$＋ｋ）＆Ｍ１（ｉ－ｘ$_{j}$＋ｋ）

上式を用いれば、情報一致判定部４２ｂでの文字列比較は、Ｃ１（ｉ＋ｋ）＆Ｍ１（ｉ＋ｋ）とＣ１（ｉ－ｘ$_{j}$＋ｋ）＆Ｍ１（ｉ－ｘ$_{j}$＋ｋ）＆Ｍ１（ｉ＋ｋ）が０≦ｋ≦Ｋ$_{j}$－１で全て一致するもののうちＫ$_{j}$が最長のものに対応するインデックスｊを求めることに帰着する。

一致探索部４２は、文字列の一致長が所定数以上であり、且つ、入力文字列と一致する文字数が最も長いものと対応する辞書インデックスを、少なくとも入力文字列に対する圧縮符号の一部として出力する。具体的には、辞書インデックスのみを出力する場合（マスクＬＺＷ法という）、辞書インデックスと、不一致と判定された最初の１文字（不一致文字という）をそのマスクデータでマスクしたマスク不一致文字Ｃ１（ｉ＋Ｋ$_{j}$）＆Ｍ１（ｉ＋Ｋ$_{j}$）とを出力する場合（マスクＬＺ７８法という）、更に辞書インデックスと連続一致文字長とマスク不一致文字Ｃ１（ｉ＋Ｋ$_{j}$）＆Ｍ１（ｉ＋Ｋ$_{j}$）とを出力する場合（マスクＬＺ７７法という）がある。更にマスクＬＺ７７法及びマスクＬＺ７８法の場合は、前記した一致探索での文字列の一致長が、所定数に達しないときには、そのことを示す符号と、その文字列の先頭の１文字のマスク不一致文字Ｃ１（ｉ）＆Ｍ１（ｉ）とが出力される。情報一致判定部４２ｂの判定結果に基づいて、符号生成部４９は、前述した各符号化方法に応じた符号を端子４４へ出力する。

【００２６】

情報一致判定部４２ｂ及び符号生成部４９は、例えば図１Ａ中の一致探索部１４及び符

号生成部１５、図２Ａ中の一致探索部３２及び符号生成部３３、図３Ａ中の一致探索部３６及び符号生成部３７とそれぞれ対応している。異なる点は、情報一致判定部４２ｂ及び符号生成部４９は、２つの文字の有効桁だけを比較して一致の判定を行うことである。同様に、ＬＺ法の前記以外の方法に、この発明を適用しても良い。その場合には、適用する対応符号化方法により処理は異なり、辞書部４３の機能構成、辞書に保持される内容、インデックスの出力等も対応する符号化方法により異なる。しかし、いずれの場合にも有効桁情報によって有効とされるビットが少なくとも一致すれば文字が一致したとして、最長の一致文字列を探索することは共通する。また、復号化の時にも、有効桁情報を使い、文字列の文字数と各文字の中の有効なビットを判別することは共通する。そして、それぞれの符号化方法に応じて復号化する。以下、この対応する符号化方法を例えばマスクＬＺ７７法、マスクＬＺ７８法、マスクＬＺＷ法、一般的にマスクＬＺ符号化方法という。

【００２７】

復号部４５は、符号生成部４９から出力された符号を文字列に復号化する。辞書登録部４６は、復号化された最長一致文字列に、最長一致文字列の最後の文字Ｃ１（ｉ＋K_j－１）の次のマスク文字（マスク不一致文字）Ｃ１（ｉ＋K_j）＆Ｍ１（ｉ＋K_j）を加えたK_j＋１文字の文字列を生成する。そして、辞書登録部４６は、辞書部４３中の現在文字列が登録されている最大のインデックスよりも１つ大きなインデックスに、生成した文字列を登録する。なお、この復号文字列の各文字は

Ｄ（ｊ）（ｋ）［ｋ：０≦ｋ＜K_j］であり、すなわち（Ｃ１（ｉ－x_j＋ｋ）＆Ｍ１（ｉ－x_j＋ｋ））［ｋ：０≦ｋ＜K_j］である。よって、辞書インデックスj_{max}に登録される新たな文字列は

（Ｃ１（ｉ－x_j＋ｋ）＆Ｍ１（ｉ－x_j＋ｋ））［ｋ：０≦ｋ＜K_j］の末尾にＣ１（ｉ＋K_j）＆Ｍ１（ｉ＋K_j）を加えたものとなる。

【００２８】

マスクＬＺ７７及びマスクＬＺ７８の場合は、復号部４５に一致長が所定値以下を表わす符号が入力されると、そのマスク不一致文字Ｃ１（ｉ）＆Ｍ１（ｉ）がそのまま辞書登録部４６へ入力され、そのマスク不一致文字が、辞書部４３中の現在文字が登録されている最大インデックスよりも１つ大きなインデックスに登録される。なお、マスクＬＺ７７及びマスクＬＺ７８の場合は、符号化開始時は、辞書部４３には、文字はなにも登録されていないが、マスクＬＺＷの場合は、符号化開始時は、辞書部４３には、全ての１文字が登録されている。復号部４５及び辞書登録部４６の機能構成及び処理は後で説明する。

また、有効桁情報は必要に応じて有効桁符号化部１７０で符号化し、有効桁符号として端子１７１より出力される。有効桁符号化部１７０では複数の文字に対する各有効桁情報をまとめて圧縮符号化して、符号化効果を高める。

【００２９】

符号化処理

実施例１の符号化処理手順の例を図６に示す。ステップＳ１では、文字（処理単位情報）Ｃ１（ｉ＋ｋ）とその有効桁情報Ｍ１（ｉ＋ｋ）が一致探索部４２に入力される（通常、プログラムの実行により符号化装置内又は外部の記憶部から取り込まれる。）。制御部４７は、全ての入力文字のステップＳ３～Ｓ６の処理が完了したかを調べる（ステップＳ２）。完了していなければ、入力文字中の最初の未処理の文字Ｃ１（ｉ）とＭ１（ｉ）に処理ポインタｉを移動させる（ステップＳ３）。マスク計算部４２ａは、Ｃ１（ｉ＋ｋ）＆Ｍ１（ｉ＋ｋ）を計算する。情報一致判定部４２ｂは、Ｃ１（ｉ＋ｋ）＆Ｍ１（ｉ＋ｋ）［ｋ：０≦ｋ］と一致するマスク文字列を、マスク計算部４８を介して辞書部４５から探索する。そして、最も長く一致する文字列Ｄ（ｊ）（ｋ）＆Ｍ１（ｉ＋ｋ）［ｋ：０≦ｋ≦K_j－１］と対応するインデックス（最長一致インデックス）j_{max}を求める（ステップＳ４）。

【００３０】

この探索は例えば図７に示すサブルーチンにより実行する（その説明は後で行う）。求めたインデックスj_{max}と対応する文字列を復号する。その復号された文字列に次の不一

致１文字$C1_D$又は次の復号先頭１文字C_Fを付加した文字列を、対応するマスクデータでマスクし、マスク文字列を生成する。そして、そのマスク文字列を辞書部４３に登録する（ステップＳ５）。マスクＬＺＷの場合は、符号生成部４９は、最長一致インデックスj_{max}を出力する。マスクＬＺ７８の場合は、符号生成部４９は、最長一致インデックスj_{max}と次のマスク不一致文字Ｃ１（ｉ＋K_{max}）＆Ｍ１（ｉ＋K_{max}）とを出力する。マスクＬＺ７７の場合は、符号生成部４９は、最長一致インデックスj_{max}と、一致文字長K_{max}と、次のマスク不一致文字Ｃ１（ｉ＋K_{max}）＆Ｍ１（ｉ＋K_{max}）とを出力する（ステップＳ６）。そして、ステップＳ２に戻る。ステップＳ２で全ての文字に対する処理が完了していれば、この入力文字列の符号化処理は終了する。ここでK_{max}は辞書のj_{max}番目のインデックスに登録されている文字列の長さ（文字数）であり、一致文字長（最長一致した文字列の文字数）を表す。

【００３１】

マスクＬＺ７７とマスクＬＺ７８の場合には、ステップＳ４で一致文字長が所定値に達しないときは、ステップＳ５でその入力文字列の先頭文字Ｃ１（ｉ）をマスクデータＭ１（ｉ）でマスクしたものを、現在文字が登録されている最大インデックスの次のインデックスに登録する。また、ステップＳ６ではそのマスク不一致文字Ｃ１（ｉ）＆Ｍ１（ｉ）を、一致文字長が所定値より少ないことを表わす符号（不一致インデックスという）と共に出力する。

【００３２】

この例では、入力文字Ｃ１（ｉ）に対し、マスクデータＭ（ｉ）で有効桁長以外のビットをマスクするマスク処理は、辞書探索の際にステップＳ４で行っているが、図６に破線で示すように文字、有効桁情報が入力されたステップ１の直後に、全ての入力文字に対して一括してマスク処理を行うこととしても良い（ステップＳ７）。また、Ｃ１（ｉ）とＣ１（ｉ）＆Ｍ１（ｉ）が必ず等しくなることがあらかじめ判っている場合には、Ｃ１（ｉ）をＭ１（ｉ）でマスクする処理は省略しても良い。

【００３３】

上述から理解されるように、ステップＳ３でのポインタｉの移動は、一致長が所定値以上の場合はｉからｉ＋k_{max}＋１となり、マスクＬＺ７７およびマスクＬＺ７８では一致長が所定値以下の場合はｉ＋１となる。マスクＬＺＷでは所定長は、辞書部４３に１つのインデックスに初期登録された文字数であり、一致長が所定値に達しない場合は生じない。

図５中に示す制御部４７がレジスタ４７ａにポインタｉを保持し、その制御と、そのポインタに基づく、Ｃ１（ｉ），Ｍ１（ｉ）の取込み、各部を動作させるための制御を行う。

【００３４】

探索処理

図７を参照して図６のステップＳ４での辞書探索処理の具体例を説明する。情報一致判定部４２ｂは、まず辞書インデックスｊを０に、最長一致文字数（長）k_{max}を０に、最長一致文字長k_{max}が得られたインデックスj_{max}を０にそれぞれ初期化する（ステップＲ１）。次に現探索インデックスｊが辞書部４３内のインデックスの最大値Ｊ以上になったかを調べる（ステップＲ２）。Ｊ以上でなければ、一致文字数ｋを０に初期化する（ステップＲ３）。辞書部４３内のｊ番目のインデックスに登録されている文字列中のｋ番目の文字Ｄ（ｊ）（ｋ－１）と入力文字列のｋ番目の文字Ｃ１（ｉ＋ｋ－１）とが、有効桁に対応するビットで一致するか否かを調べる。この例では、辞書部４３のインデックスｊに登録されたｋ番目の文字Ｄ（ｊ）（ｋ－１）と入力文字列のｋ番目の文字Ｃ１（ｉ＋ｋ－１）を、マスクデータＭ１（ｉ＋ｋ－１）を用いて比較する。具体的には、まず辞書文字Ｄ（ｊ）（ｋ－１）とマスクデータＭ１（ｉ＋ｋ－１）とのビットごとの論理積をとる（ステップＲ１０）。その後、インデックスｊのｋ番目のマスク辞書文字Ｄ（ｊ）（ｋ－１）＆Ｍ１（ｉ＋ｋ－１）と、ポインタｉのｋ番目のマスク入力文字Ｃ１（ｉ＋ｋ－１）

＆Ｍ１（ｉ＋ｋ－１）とが等しいか否かを調べる。（ここで、マスク入力文字Ｃ１（ｉ＋ｋ－１）＆Ｍ１（ｉ＋ｋ－１）を計算する処理は、ステップＲ１０で行うこととしても良いし、図６中に破線で示したステップＳ７で事前に計算しておいても良い。）

【００３５】

両マスク文字が一致するかを確認する（ステップＲ４）。一致する場合は、ｋに１を加えてステップＲ１０に戻る（ステップＲ５）。ステップＲ４で両者が一致しなければ、一致文字数ｋが現在の最長一致文字長k_{max}より大きいか否かを調べる（ステップＲ６）。ｋの方が大きければ最長インデックスj_{max}を、現在のインデックスｊに更新し、最長一致文字長k_{max}を、現在のｋに更新する（ステップＲ７）。ステップＲ６でｋの方が大きくない場合またはステップＲ７の処理が終わった場合には、辞書インデックスｊに１を加えてステップＲ２に戻る（ステップＲ８）。このようにして各インデックスｊの登録マスク文字列と入力マスク文字列とを、マスクをかけた状態で比較し、不一致文字が生じるまで一文字ずつ順次調べる。そして、ステップＲ２でｊがインデックスの最大値Ｊ以上になると、その時の最長一致文字数k_{max}が得られた時のインデックスj_{max}を出力する（ステップＲ９）。この辞書探索処理は、マスクＬＺ７７、マスクＬＺ７８、マスクＬＺＷのいずれに対しても同様に行うことができる。

【００３６】

k_{max}が所定値以上の場合は、圧縮符号化装置から出力される符号は、マスクＬＺ７７及びマスクＬＺ７８では最長一致インデックスj_{max}と、一致文字長k_{max}（マスクＬＺ７８ではk_{max}は省略される。）と、その時の処理ポインタｉの入力文字列中の最初に不一致となった１文字Ｃ１（ｉ＋k_{max}）（一般に$C1_D$と表わす）とそのマスクデータＭ１（ｉ＋k_{max}）（一般に$M1_D$と表わす）とのビットごとの論理積（$C1_D$＆$M1_D$）である。また、k_{max}が所定値より小さい場合は、圧縮符号化装置から出力される符号は、不一致を表わすインデックス（例えば背景技術中での説明ではインデックスが０、一致文字数が０：以下不一致インデックスという。）と、その処理ポインタｉの先頭入力文字Ｃ１（ｉ）（不一致入力１文字$C1_D$という。）とそのマスクデータＭ１（ｉ）との論理積（Ｃ１（ｉ）＆Ｍ１（ｉ））（マスク不一致文字という。）である。

【００３７】

従って図５中に示すように、図１Ａ中の符号生成部１５、図２Ａ中の符号生成部３３と対応する符号生成部４９が設けられる。マスクＬＺＷの場合はj_{max}のみが出力される。図７に示した処理が可能のために、例えば図５の情報一致判定部４２ｂには、前記パラメータｊ、k_{max}，j_{max}，ｋがそれぞれ格納されるレジスタ$42b_A$，$42b_B$，$42b_C$，$42b_D$が設けられ、ステップＲ２，Ｒ４及びＲ６の判定を行う判定部$42b_E$が設けられ、処理手順を制御する制御部$42b_F$が設けられる。この制御部$42b_F$は制御部４７で兼用してもよい。

【００３８】

実施例１では、辞書部４３内の登録文字（情報）と入力文字との比較は、文字と辞書登録文字とを、共にマスクした状態で一致するかを調べている。したがって、マスクされている部分に不一致のビットが存在していても、両文字は一致したと判定され、結果として文字列も一致と処理される。このため、最長一致インデックス（j_{max}）が出力される回数が従来方法より多くなり、出力符号の圧縮率を向上させることができる。このように、マスク文字の一致Ｃ１（ｉ＋ｋ）＆Ｍ１（ｉ＋ｋ）＝Ｄ（ｊ）（ｋ）＆Ｍ１（ｉ＋ｋ）を検出する処理は、全ての方式で同じである。そして、マスクＬＺＷでは、一致文字数が最大のインデックスj_{max}を符号として出力する。一致文字数が所定数より多ければ、マスクＬＺ７８では、j_{max}と$C1_D$＆$M1_D$を符号として出力する。マスクＬＺ７７では、j_{max}とk_{max}と$C1_D$＆$M1_D$を符号として出力する。また、一致文字数が所定数より少なければ、マスクＬＺ７７とマスクＬＺ７８では、一致文字数が所定数より少ないことを表わすインデックスとＣ１（ｉ）＆Ｍ１（ｉ）を符号として出力する。

【００３９】

ＬＺ７７の場合は、出力符号を復号部４５で復号して辞書部４３に登録する。一致文字

数が所定以上の場合には、辞書部４３には、符号出力された最長一致インデックスj_{max}に対応する文字列Ｄ（j_{max}）（０）～Ｄ（j_{max}）（$k_{max}-1$）にマスク不一致文字Ｃ１（ｉ＋k_{max}）＆Ｍ１（ｉ＋k_{max}）を付加した文字列が、スライド辞書の最も新しい文字列の格納位置に登録される。一致文字数が所定数より少ない場合には、辞書部４３には、マスク文字Ｃ１（ｉ）＆Ｍ１（ｉ）が、スライド辞書の最も新しい文字列の格納位置に登録される。

ＬＺ７８、ＬＺＷの場合は、符号出力された最長一致インデックスj_{max}に対応する文字列Ｄ（j_{max}）（０）～Ｄ（j_{max}）（$k_{max}-1$）にマスク不一致文字Ｃ１（ｉ＋k_{max}）＆Ｍ１（ｉ＋k_{max}）を付加した文字列が、辞書部４３のインデックスＪの位置に登録される。登録の後、Ｊは１増加される。

【００４０】

復号化構成１

実施例１の復号化装置を説明する。図８に、マスクＬＺ７７、マスクＬＺ７８に適用される復号化装置を示す。この構成は、図５に示した符号化装置中の復号部４５として用いることもできる。

符号解析部５２は、入力端子５１からの入力符号を、最長一致インデックスj_{max}と、一致文字長k_{max}（マスクＬＺ７８では省略されている）と、マスク不一致文字（$C1_D$＆$M1_D$）とに分離する。インデックスj_{max}と一致文字長k_{max}が情報取得部５３に入力される。マスクＬＺ７７の場合は、情報取得部５３は、辞書部５４のインデックスj_{max}の位置から、一致文字長k_{max}分の文字列Ｄ（j_{max}）（ｋ）［ｋ：０≦ｋ＜k_{max}］を取り出す。マスクＬＺ７８の場合は、情報取得部５３は、辞書部５４のインデックスj_{max}に登録されているk_{max}文字の文字列Ｄ（j_{max}）（ｋ）［ｋ：０≦ｋ＜k_{max}］を取り出す。そして、マスク計算部１５２と連結部１５３に渡す。マスク計算部１５２は、インデックスj_{max}を復号して得た文字列のｋ番目の文字と有効桁情報Ｍ１（ｉ＋ｋ－１）との論理積演算によって、文字列（Ｄ（j_{max}）（ｋ）＆Ｍ１（ｉ＋ｋ））［ｋ：０≦ｋ＜k_{max}］を生成し、文字列合成部５５に入力する。文字列合成部５５は、入力された文字列の最後に、符号解析部５２からのマスク不一致文字（$C1_D$＆$M1_D$）を連結して、出力端子５６へ復号文字列として出力する。また、連結部１５３は、情報取得部５３から渡された文字列Ｄ（j_{max}）（ｋ）［ｋ：０≦ｋ＜k_{max}］に、符号解析部５２からのマスク不一致文字（$C1_D$＆$M1_D$）を連結して辞書登録部５７に渡す。辞書登録部５７は、連結部１５３から入力された文字列を辞書部５４に登録する（辞書に登録する文字列Ｄ（j_{max}）（ｋ）［ｋ：０≦ｋ＜k_{max}］にはマスクをかけない。）。符号解析部５２から一致文字長が所定値より小さいことを示す符号（不一致インデックス）が検出された場合は、入力されたマスク不一致文字$C1_D$＆$M1_D$を、復号文字として出力端子５６へ出力すると共に辞書登録部５７を用いて辞書部５４に登録する。この登録処理は、図１Ｂ中の辞書登録部２７と同じである。なお有効桁符号が入力される場合は、その有効桁符号は有効桁復号化部１５６で、有効桁情報群を復号する。そして、その有効桁情報群から、文字Ｄ（j_{max}）（ｋ）と対応する有効桁情報Ｍ１（ｉ＋ｋ）を順次マスクデータ生成部１５１に入力する。

【００４１】

復号化処理１

図８に示した機能構成の処理手順例を、図９を参照して説明する。符号解析部５２が符号を受信する（ステップＳ１１）。制御部１５７は、全ての符号に対するステップＳ１３～Ｓ１９の処理が完了したかを調べる（ステップＳ１２）。完了していなければ、処理ポインタi_Cを移動する（ステップＳ１３）。制御部１５７は、そのポインタi_Cが指す未処理の入力された符号Ｂ１（i_C）中のインデックスが、不一致インデックスか否かを判定する（ステップＳ１４）。不一致インデックスでなければ、情報取得部５３は、そのインデックスj_{max}が示す辞書部５４中のインデックスj_{max}に登録されている文字列を取得する（ステップＳ１５）。マスクＬＺ７７の場合は、インデックスが示す位置から、未処理の入力された符号Ｂ１（i_C）中の一致文字長k_{max}分の文字列を取得する。マスク

ＬＺ７８の場合は、インデックスに対応する位置に登録されているk_{max}文字の文字列を取得する。

【００４２】

取得した文字列に、有効桁情報Ｍ１（i_C＋ｋ－１）を論理積演算して、文字列（Ｄ（j_{max}）（ｋ）＆Ｍ１（ｉ＋ｋ））［ｋ：０≦ｋ＜k_{max}］を生成する（ステップＳ１６）。このマスク文字列にマスク不一致１文字を連結して復号文字列として出力する（ステップＳ１７）。

また、ステップＳ１５で取得した文字列（マスクをかけていない文字列）Ｄ（j_{max}）（ｋ）［ｋ：０≦ｋ＜k_{max}］に、マスク不一致の１文字を連結して辞書部５４に登録してステップＳ１２に戻る（ステップＳ１８）。

【００４３】

ステップＳ１４で不一致インデックスであると判定されると、符号解析部５２で分離されたマスク不一致１文字を復号文字として出力する。さらにそのマスク不一致１文字を辞書部５４に登録してステップＳ１２に戻る（ステップＳ１９）。ステップＳ１２で全ての符号に対する処理が完了したらその入力符号列に対する処理を終了する。図８中の制御部１５７は各部を順次動作させ、また処理パラメータi_Cが格納されるレジスタ１５７ａ、ステップＳ１２、Ｓ１９の各判定を行う判定部１５７ｂを備えている。

【００４４】

復号化構成２

実施例１でのマスクＬＺＷに対する復号化装置の例を図１０に示す。この場合は入力端子５１に入力される文字（処理単位情報）列はインデックスj_{max}のみである。情報取得部５３は、辞書部５８中に登録されているそのインデックスj_{max}に対応する文字列Ｄ（j_{max}）（ｋ）［ｋ：０≦ｋ＜k_{max}］を取り出す。マスクデータ生成部１５１は、それぞれの文字Ｄ（j_{max}）（ｋ）に対応した有効桁情報Ｍ１（ｉ＋ｋ）を生成する。マスク計算部１５２は、有効桁情報Ｍ１（ｉ＋ｋ）と対応文字Ｄ（j_{max}）（ｋ）との論理演算を行って文字列（Ｄ（j_{max}）（ｋ）＆Ｍ１（ｉ＋ｋ））［ｋ：０≦ｋ＜k_{max}］を生成し、復号文字列として出力端子５６に出力する。有効桁情報が有効桁符号として端子５１’に入力された場合は、図中に破線で示すように、有効桁復号化部１５６で復号化して有効桁情報を得る。

【００４５】

また、情報取得部５３で取得した文字列は、辞書登録部５９にも入力される。新たな文字列が復号化され出力されると、辞書登録部５９は、直前バッファ５９ａに一時的に保持されている直前に辞書部から取り出された文字列Ｄ（j_{max}’）（ｋ）［ｋ：０≦ｋ＜k_{max}’］の後に、新たに復号化された（辞書部から取り出された）文字列の先頭文字C_F＝Ｄ（j_{max}）（０）を連結して辞書部５８に登録する。そして、辞書登録部５９は、復号文字列Ｄ（j_{max}）（ｋ）［ｋ：０≦ｋ＜k_{max}］を直前バッファ５９ａに一時的に保持する。

【００４６】

復号化処理２

このマスクＬＺＷの復号化処理手順の例を図１１に示す。まず、情報取得部５３は、符号入力を取り込む（ステップＳ１１）。制御部１５７は、全符号のステップＳ１３～Ｓ１９の処理が完了したかを調べる（ステップＳ１２）。完了していなければ処理ポインタi_Cを移動させる（ステップＳ１３）。ステップＳ１１～Ｓ１３までは、図９に示した場合と同じである。マスクＬＺＷでは、次に情報取得部５３は、符号Ｂ１（i_C）のインデックスj_{max}に対応する文字列Ｄ（j_{max}）（ｋ）［ｋ：０≦ｋ＜k_{max}］を辞書部５８から取り出す。辞書登録部５９は、直前バッファ５９ａに一時的に文字列Ｄ（j_{max}）（ｋ）［ｋ：０≦ｋ＜k_{max}］を保持する（ステップＳ１７）。それぞれの文字Ｄ（j_{max}）（ｋ）に対応した有効桁情報Ｍ１（ｉ＋ｋ）とＤ（j_{max}）（ｋ）との論理積演算によって文字列（Ｄ（j_{max}）（ｋ）＆Ｍ１（ｉ＋ｋ））［ｋ：０≦ｋ＜k_{max}］を生成し、復号文字列として出力端子５６から出力する（ステップＳ１８）。

【００４７】
新たな文字列が復号化され出力されると、辞書登録部５９は、直前バッファ５９ａに一時的に保持していた直前に辞書部から取り出された文字列Ｄ（j_{max}'）（ｋ）［ｋ：０≦ｋ＜k_{max}'］の後に、新たに復号化された（辞書部から取り出された）文字列の先頭文字C_F＝Ｄ（j_{max}）（０）を連結して、辞書部５８に登録する。そして、辞書登録部５９は、復号文字列Ｄ（j_{max}）（ｋ）［ｋ：０≦ｋ＜k_{max}］を直前バッファ５９ａに一時的に保持してステップＳ１２に戻る（ステップＳ１９）。
以上述べたことから理解されるように、実施例１の復号化装置は、従来のＬＺ７７，ＬＺ７８，ＬＺＷの各方式に対する復号化装置と、辞書部の構成、マスク計算を行う部分および辞書に登録する内容が異なる。

【００４８】
［実施例１変形］
以下に実施例１の各種変形例を説明する。
先に述べた実施例１の説明に用いた具体例を以下実施例Ａ－３という。この実施例Ａ－３では、マスクＬＺ７７、マスクＬＺ７８の場合は、図５中に示したように、入力文字列および辞書に登録された文字列の双方にマスクＭ１（ｉ＋ｋ）をかけた状態で文字列の一致判定を行う。また、マスク計算部４２ａが、不一致文字$C1_D$を、そのマスクデータ$M1_D$によりマスク処理したマスク不一致文字$C1_D$＆$M1_D$、つまり不一致文字中の有効でない桁（ビット）部分は固定値（前記例では“０”であるが“１”でもよい）としたものの情報を求める。そして、最長一致した文字列を示す辞書インデックスj_{max}と、マスク不一致文字$C1_D$＆$M1_D$とを符号として出力する。復号部４５は、辞書部４３の辞書インデックスj_{max}に対応する文字列Ｄ（j_{max}）（ｋ）［ｋ：０≦ｋ＜k_{max}］を取り出す。また、辞書登録部４６が、マスク不一致文字$C1_D$＆$M1_D$を連結した文字列を辞書部４３に登録する。

【００４９】
辞書探索では、Ｃ１（ｉ＋ｋ）＆Ｍ１（ｉ＋ｋ）＝Ｄ（ｊ）（ｋ）＆Ｍ１（ｉ＋ｋ）、［ｋ：０≦ｋ＜K_j］となる最長のK_jを与える辞書インデックスｊを探した。ここで、Ｄ（ｊ）（ｋ）に登録されている内容はx_j文字数だけ過去に入力された文字列である。したがって、その文字列を構成する各文字は前述したようにＤ（ｊ）（ｋ）＝Ｃ１（ｉ－x_j＋ｋ）＆Ｍ１（ｉ－x_j＋ｋ）とあらわすこともできる。
すなわち、辞書探索とは、Ｃ１（ｉ＋ｋ）＆Ｍ１（ｉ＋ｋ）＝Ｃ１（ｉ－x_j＋ｋ）＆Ｍ１（ｉ－x_j＋ｋ）＆Ｍ１（ｉ＋ｋ）、［ｋ：０≦ｋ＜K_j］となる最長のK_jを与える辞書インデックスｊを探索することに相当する。
また、復号側では、文字列Ｄ（j_{max}）（ｋ）＆Ｍ１（ｉ＋ｋ）［ｋ：０≦ｋ＜k_{max}］にマスク不一致文字$C1_D$＆$M1_D$を連結した文字列を出力する。
以下にこの実施例Ａ－３に対する変形実施例を、主として実施例Ａ－３と異なる点について説明する。

【００５０】
実施例Ａ－１
この実施例Ａ－１では、入力文字列および辞書に登録された文字列の双方にマスクＭ１（ｉ＋ｋ）をかけた状態で文字列の一致判定を行う。そして、最長一致した辞書インデックスj_{max}と、マスクをかけない不一致文字$C1_D$そのものの情報を符号として出力する。このため図５中に破線１４１で示すように文字Ｃ１（ｉ＋ｋ）も符号生成部４９に入力される。そして、符号生成部４９は、文字Ｃ１（ｉ＋ｋ）を不一致文字$C1_D$として出力する（Ａ－３ではマスク不一致文字$C1_D$＆$M1_D$を出力）。

【００５１】
復号部４５は、辞書部４３から辞書インデックスj_{max}に対応する文字列Ｄ（j_{max}）（ｋ）［ｋ：０≦ｋ＜k_{max}］を取り出す。辞書登録部４６は、不一致文字$C1_D$そのものを連結した文字列を、辞書部４３に登録する。したがって、辞書部４３にはマスク処理されていない文字列が登録されることになる。

つまり、辞書探索では、Ｃ１（ｉ＋ｋ）＆Ｍ１（ｉ＋ｋ）＝Ｄ（ｊ）（ｋ）＆Ｍ１（ｉ＋ｋ）、［ｋ：０≦ｋ＜K_j］となる最長のK_jを与える辞書インデックスｊを探す。ここで、Ｄ（ｊ）（ｋ）に登録されている内容はx_j文字だけ過去に入力された文字列に対応する文字列であり、その文字列の各文字はＤ（ｊ）（ｋ）＝Ｃ１（ｉ－x_j＋ｋ）とあらわすこともできる。

【００５２】

すなわち、辞書探索とは、Ｃ１（ｉ＋ｋ）＆Ｍ１（ｉ＋ｋ）＝Ｃ１（ｉ－x_j＋ｋ）＆Ｍ１（ｉ＋ｋ）、［ｋ：０≦ｋ＜K_j］となる最長のK_jを与える辞書インデックスｊを探索することに相当する。

また、復号側では、マスク計算部１５２の出力ではなく図８に破線１５７′で示すように辞書部５４から取得した文字列Ｄ（j_{max}）（ｋ）［ｋ：０≦ｋ＜k_{max}］に、図８中に括弧書きで示すように符号解析部５２から取り出した不一致文字$C1_D$を連結した文字列を出力する。

【００５３】

この実施例Ａ－１では、辞書部４３内に登録される文字はマスクされていない。しかし、入力文字列と辞書に記録された文字列（辞書文字列）とは、共にマスクをかけた状態で比較され、一致長が最も長い辞書文字列が選択される。このため、最長一致インデックスj_{max}が複数得られる場合がある。この場合、同一のj_{max}を得る辞書文字列の中で、入力文字列との距離（例えば数値の場合、両数値の差）が最も小さい辞書文字列を選択する。これは、入力文字列に最も近い辞書文字列を選択することである。

【００５４】

例えば、図７中のステップＲ６において、ｋがk_{max}より大きくないと判定されると、図１２Ａに示すように、ｋがk_{max}と等しいかを調べる（ステップＲ１２）。等しければ、その時の入力文字列Ｃ１（ｉ＋ｎ）［ｎ：０≦ｎ≦ｋ－１］と辞書文字列Ｄ（ｊ）（ｎ）［ｎ：０≦ｎ≦ｋ－１］との距離を計算する。そして、その距離をj_{max}と対応させて記憶部に格納し、ステップＲ８へ移る（ステップＲ１３）。ステップＲ１２でｋがk_{max}と等しくなければ、ステップＲ８へ移る。これらの機能構成として、例えば図１２Ｂに示すように、図５中の情報一致判定部４２ｂと符号生成部４９との間に評価部４２ｃを設ける。評価部４２ｃは、ステップＲ１２を実行する判定部$42C_A$、ステップＲ１３を実行する距離計算部$42C_B$、各j_{max}と距離との対応を記憶する記憶部$42C_C$、記憶部$42C_C$中の複数のj_{max}の中で距離が最も短かいものを選択する選択部$42C_D$が設けられる。なお、図７中のステップＲ７のj_{max}更新は、記憶部$42C_C$中のj_{max}に対しても行われる。

【００５５】

実施例Ａ－１の説明は、マスクＬＺ７７またはマスクＬＺ７８の場合である。マスクＬＺＷの場合は、図１０中のマスク計算部１５２及び図１１中のステップＳ１８でのマスク処理は不要となる。

【００５６】

実施例Ａ－１′

実施例Ａ－１では、復号側は、マスク処理することなくＤ（j_{max}）（ｋ）［ｋ：０≦ｋ＜k_{max}］と$C1_D$を出力した。実施例Ａ－１′では、マスク計算部１５８（図８中に破線で示す。）が、符号解析部５２よりの不一致文字$C1_D$（図８中に括弧書きで示す。）に、マスクデータ$M1_D$でマスク処理したマスク不一致文字$C1_D$＆$M1_D$を計算する。文字列合成部５５は、マスク計算部１５２からのマスク文字列Ｄ（j_{max}）（ｋ）＆Ｍ１（ｉ＋ｋ）［ｋ：０≦ｋ＜k_{max}］に、マスク計算部１５８からのマスク不一致文字$C1_D$＆$M1_D$を連結した文字列を出力する。

なお、マスクＬＺＷの場合は、図１０及び図１１を参照した実施例Ａ－３の処理と同様である。

【００５７】

実施例Ａ－２

実施例Ａ－３と異なる点は、図５の破線１４２で示すように、辞書部４３から取り出した文字Ｄ（ｊ）（ｋ）を情報一致判定部４２ｂに直接入力する。つまり、マスク計算部４８を用いない。辞書部４３に登録された文字列は、実施例Ａ－３と同様に、マスクされたものである。従って情報一致判定部４２ｂでは、Ｃ１（ｉ＋ｋ）＆Ｍ１（ｉ＋ｋ）＝Ｄ（ｊ）（ｋ）が成立すると両文字が一致したと判定する。

復号化側では、図８の破線１５７′で示すように、情報取得部５３が取得したマスク文字列が、マスク計算部１５２を介することなく文字列合成部５５に入力される。

【００５８】

実施例Ａ－４

実施例Ａ－４は、入力文字Ｃ１（ｉ＋ｋ）と辞書文字Ｄ（ｊ）（ｋ）との有効桁長の短い方に合わせて一致するインデックスを探索する点が実施例Ａ－３と異なる。

辞書部４３は、図１３Ａに示すように、マスクされていない文字列Ｄ（ｊ）（ｋ）[ｋ：０≦ｋ＜K_j]が格納される文字部４３ａと、各文字列の各文字の有効桁情報（例えばマスクデータM_D（ｊ）（ｋ）[ｋ：０≦ｋ＜K_j]）が格納されている有効桁部４３ｂとを有する。マスク計算部４８′には、入力文字Ｃ１（ｉ＋ｋ）のマスクデータＭ１（ｉ＋ｋ）と、辞書部４３から辞書文字Ｄ（ｊ）（ｋ）に対応するマスクデータM_D（ｊ）（ｋ）とが入力される。マスク計算部４８′は、一致判定に用いるマスクデータＭ’を、入力されたマスクデータのビットごとの論理積によって計算する。すなわち、マスク計算部４８′は、Ｍ’（ｋ）＝Ｍ１（ｉ＋ｋ）＆M_D（ｊ）（ｋ）を計算する。また、マスク計算部４２ａはマスク入力文字Ｃ１（ｉ＋ｋ）＆Ｍ’（ｋ）を、マスク計算部４８′はＤ（ｊ）（ｋ）＆Ｍ’（ｋ）をそれぞれ計算する。そして、Ｃ１（ｉ＋ｋ）＆Ｍ’（ｊ）（ｋ）とＤ（ｉ）＆Ｍ’（ｋ）とが、情報一致判定部４２ｂに入力される。

【００５９】

符号出力としては、最長一致した文字列の辞書インデックスj_{max}（マスクＬＺ７７の場合は、さらに一致長k_{max}）は、同様に出力される。しかし、不一致となった文字$C1_D$は、マスクされたものではなく、不一致文字$C1_D$自体が出力される。

入力文字Ｃ１（ｉ＋ｋ）の有効桁長が辞書文字Ｄ（ｊ）（ｋ）の有効桁長より長い場合は、更に次のデータをも出力する必要がある。例えば図５中の符号生成部４９内に処理部４９ａを設ける。処理部４９ａの機能構成を図１３Ｂに示す。比較部４９ｂは、マスクデータを２進数値と見なした時のＭ１（ｉ＋ｋ）とＭ’（ｋ）とを構成するビットごとに比較する。ここで、Ｍ１（ｉ＋ｋ）が１、且つ、Ｍ’（ｋ）が０となるビットがある場合には、入力文字Ｃ１（ｉ＋ｋ）のそのビットに対応する情報を、別に出力する必要がある。そこで、演算部４９ｃは、Ｍ１（ｉ＋ｋ）とＭ’（ｋ）とのビットごとの排他的論理和を演算する。その演算結果Ｍ’’（ｋ）＞０となった場合には、演算部４９ｄは、Ｍ’’（ｋ）の１となったビットに対応する入力文字Ｃ１（ｉ＋ｋ）のビットを取り出し、結果を出力する。

【００６０】

このようにしてＣ１（ｉ＋ｋ）中の有効桁（ビット）であるが、Ｄ（ｊ）（ｋ）中の対応する桁（ビット）が有効桁でない部分のビットが、j_{max}、不一致文字$C1_D$と共に出力される。

マスクＬＺＷ方法の場合は、符号出力の際に図１３Ｂと対応する処理が行われるが、不一致１文字$C1_D$は出力されない。

復号化装置では、図８中に括弧書きで示し、かつ１点鎖線で示すように、情報取得部５３から取り出された文字列Ｄ（j_{max}）（ｋ）［ｋ：０≦ｋ＜k_j］の各ｋに対して、Ｍ１（ｉ＋ｋ）とM_D（j_{max}）（ｋ）を用いてビットを補ってから出力すべきビットを算出する。すなわち、比較部１５４は、Ｍ１（ｉ＋ｋ）＾（Ｍ１（ｉ＋ｋ）＆M_D（j_{max}）（ｋ））＞０となるｋを求める。そして、求めたｋに対して、Ｍ１（ｉ＋ｋ）＾（Ｍ１（ｉ＋ｋ）＆M_D（j_{max}）（ｋ））で１となっているビットに対応する情報を符号解析部５２から取り出す。演算部１５５は、Ｍ１（ｉ＋ｋ）＾（Ｍ１（ｉ＋ｋ）＆M_D（j_{max}）（ｋ））で１となっているビットを、符号解析部５２から取り出したビット情報と置

き換える。そうして得られたマスク情報がＭ１’（ｉ＋ｋ）としてマスクデータ生成部１５１に入力される。そして、マスク計算部１５２は、マスク処理によって出力すべき文字列を復元する。即ち（Ｄ（j_{max}）（ｋ）＆M_D（j_{max}）（ｋ）＆Ｍ１（ｉ＋ｋ））｜Ｍ１’（ｉ＋ｋ）が計算される。なお、Ａ＾ＢはＡとＢのビットごとの排他的論理和を表す。Ａ｜ＢはＡとＢの論理和を表す。

【００６１】

また、上記の処理によって得られた文字列に、マスクされない不一致文字$C1_D$が文字列合成部５５で合成されることになる。

辞書部への登録は、情報取得部５３で取得された文字列Ｄ（j_{max}）（ｋ）［ｋ：０≦ｋ＜k_j］とその有効桁情報Ｍ（j_{max}）（ｋ）［ｋ：０≦ｋ＜k_j］及び不一致文字$C1_D$とその有効桁情報$M1_D$となる。

【００６２】

実施例Ａ－４’

実施例Ａ－４’では、不一致文字$C1_D$に対しそのマスクデータ$M1_D$でマスクして出力する。その他は実施例Ａ－４と同様である。

【００６３】

実施例Ａ－５

実施例Ａ－５は、実施例Ａ－４との違いのみを説明する。辞書部４３には文字Ｄ（ｊ）（ｋ）ではなく、そのマスクした文字Ｄ（ｊ）（ｋ）＆M_D（ｊ）（ｋ）が登録される点、符号化出力中の不一致文字$C1_D$がマスク文字$C1_D$＆$M1_D$として出力される点が異なる。復号化装置は実施例Ａ－４と同じである。

【００６４】

実施例Ａ－１～Ａ－３では、例えばマスクＬＺ７７を例とすると、辞書部４３には、図１４Ａに示すように、辞書部４３に最後に入力された所定量の文字列が登録されている。図１４Ａでは、辞書部４３は、構成するスライドバッファと繋がった先読みバッファ６５を有する。バッファ６５の文字部６５ａに次に符号化されるべき文字を先頭とする文字列が格納される。また先読みバッファ６５内の各文字Ｃ１（ｉ＋ｋ）の有効桁情報（この例ではマスクデータＭ１（ｉ＋ｋ））が有効桁部６５ｂに格納される。

実施例Ａ－４～Ａ－５では、図１４Ｂに示すように、辞書部４３は、図１４Ａ中の辞書部４３と同一内容を登録する文字部４３ａと、文字部４３ａに登録されている各文字Ｄ（ｊ）（ｋ）の有効桁情報（この例ではマスクデータM_D（ｊ）（ｋ））を格納する有効桁部４３ｂを有する。先読みバッファ６５は、実施例Ａ－１～Ａ－３の場合と同様である。なおマスクＬＺ７８、マスクＬＺＷの場合は、辞書部４３はスライドバッファではなく固定バッファとなる。

【００６５】

実施例Ａ－１，Ａ－１’，Ａ－２，Ａ－３，Ａ－４，Ａ－４’，Ａ－５の順で、辞書登録文字との一致が得られ易い。従って、同じインデックスj_{max}が出力される回数が多くなり、圧縮効率が向上する。しかし、インデックスj_{max}の探索処理が多くなる。また実施例Ａ－４，Ａ－４’，Ａ－５では、Ｍ１（ｉ＋ｋ）＾（Ｍ１（ｉ＋ｋ）＆M_D（ｊ）（ｋ））＞０の場合には、入力文字は有効桁のみに意味があるが、辞書文字としては有効桁でないビット（列）も出力することになる。したがって、圧縮効率が低下する。従って、信号の性質により探索処理数（時間）や、圧縮効率を考慮して、いずれの実施例を用いるかを選択すればよい。

【００６６】

［実施例２］

実施例２は、音声や音楽などのオーディオ信号、画像信号、各種計測信号、線形予測誤差信号などのディジタル値のサンプル列の圧縮符号化、復号化に実施例１に示した方法、装置を適用したものである。実施例１では各入力文字（情報）は一定のビット長であり、かつその有効桁情報が既知であることを前提とした。本実施例では、オーディオ信号などのディジタル値のサンプル列を、実施例１を適用できるように、所定ビット数の文字（情

報）からなる処理情報列の単位に変換する処理について主に説明する。

【００６７】

まず、そのような変換を行う理由を説明する。前記サンプル列の各サンプルの各ビット長は様々であり、一定ビット長の文字（情報）の文字列（単位処理情報列）ではない場合がある。そこで、次のような変換方法が考えられる。

各サンプルを構成するビットを、１ビット単位に分解し、順に並べたビット列に変換する。そして、処理の単位となる１文字長（たとえば４ビット）ごとに切り出す。切り出した１文字長の文字列の繰り返しを、情報圧縮符号化装置の入力とする。

オーディオ信号などの波形信号のサンプル列は、本来、隣接するサンプル間の相関（冗長性）や、１サンプル内での相関（冗長性）などが大きいという性質を持っている。しかし、前記のような変換方法では、元のサンプル列が有していた性質が損なわれる場合が多い。

本発明は、サンプル間及び／又はサンプル内の相関（冗長性）も情報列の圧縮に有効に利用するために、サンプル列を相関（冗長性）が保持されるように単位処理情報列に変換する。まず、サンプル列を単位処理情報（文字）に変換した具体例を示す。

【００６８】

単位処理情報変換例１（可変長サンプル）

図１５Ａに可変長、つまりサンプルごとに語長（ビット数）が異なる場合の入力サンプル列の例を示す。入力サンプル列（ｓ１，ｓ２，…）中の各サンプルの語長（ビット数）L_Sが可変長の場合、このままでは単位処理情報のビット数が一定（固定文字サイズ）であることが必要なので、実施例１のマスクＬＺ符号方法を適用することができない。そこで、例えばマスクＬＺ法の単位処理情報の語長（ビット数）N_Cを、入力サンプルの最大語長（ビット数）$L_{S\ max}$以上にとって単位処理情報とする。

【００６９】

図１５Ｂは、各入力サンプルをＬＳＢ（最下位ビット）側に整列し、単位処理情報へ変換した例を示す図である。各入力サンプルｓ１，ｓ２，…は、ＬＳＢ側に整列され、有効桁よりもＭＳＢ側にはダミービット（例えば０）を付加して語長（ビット数）を拡張される。このようにして、各入力サンプルを、同一ビット数（単位処理情報のビット数N_C）に揃える。

図１５Ｃは、各入力サンプルをＭＳＢ（最上位ビット）側に整列し、単位処理情報へ変換した例を示す図である。各入力サンプルｓ１，ｓ２，…は、ＭＳＢ側に整列され、有効桁よりもＬＳＢ側にはダミービット（例えば０）を付加して語長（ビット数）を拡張される。このようにして、各入力サンプルを、同一ビット数（単位処理情報のビット数N_C）に揃える。

【００７０】

入力サンプルの最大語長$L_{S\ max}$よりも短い語長（ビット数）をマスクＬＺ法の単位処理情報のビット数（１文字のサイズ）として変換を行ってもよい。図１５Ｄは、各入力サンプルをＬＳＢ（最下位ビット）側に整列し、単位処理情報（１文字）のビット数（語長）N_Cずつに切り出した例である。この例では、各入力サンプルｓ１，ｓ２，…をＬＳＢ側に整列し、ＬＳＢ側からＭＳＢ側に向かってマスクＬＺ処理の１単位処理情報（１文字）のビット数（語長）N_Cずつ切り出して単位処理情報（１文字）に分割する。この際、入力サンプルの語長L_Sが単位処理情報の語長N_Cの整数倍でない場合には、余りのビットが生じる。この余りのビットには、ＭＳＢ側に単位語長N_Cになるまでダミービット（例えば０）を付加して単位処理情報に変換する。サンプルｓ１のビット長L_Sは、N_Cより大で$2N_C$より小さい。そこで、最初のN_Cビットを切り出し、残りのビットのＭＳＢ側にダミービットを付加することで、ビット数がN_Cの単位処理情報とする。なお、単位語長N_Cの整数倍よりもＭＳＢ側へのダミービット付加は行わない。このように切り出された単位処理情報（文字）は、最初のサンプルｓ１のＬＳＢ側からＭＳＢ側、次のサンプルｓ２のＬＳＢ側からＭＳＢ側のように整列される。図１５Ｄでは、単位処理情報の整列順に、番号１、２、３、…、１２を付けてある。この例では２文字（単位処理情報）目、

５文字（単位処理情報）目、１２文字（単位処理情報）目にダミービットが付加されている。なお、辞書登録文字（単位処理情報）との探索比較を行う際にはダミービット部分はマスクされる。

【００７１】

図１５Ｅは、各入力サンプルをＭＳＢ（最上位ビット）側に整列し、単位処理情報（１文字）のビット数（語長）N_Cずつに切り出した例である。この例では、各入力サンプルｓ１，ｓ２，…をＭＳＢ側に整列し、ＭＳＢ側からＬＳＢ側に向かって単位処理情報（１文字）の語長N_Cずつ切り出して１文字（単位処理情報）に分割する。この際、入力サンプルの語長L_Sが単位処理情報の語長N_Cの整数倍でない場合には、余りのビットが生じる。この余りのビットには、ＬＳＢ側に単位処理情報の語長N_Cになるまでダミービットを付加する。単位語長N_Cの整数倍よりもＬＳＢ側へのダミービット付加は行わない。各文字（単位処理情報）は、最初のサンプルｓ１のＭＳＢ側からＬＳＢ側、次のサンプルｓ２のＭＳＢ側からＬＳＢ側のように整列される。図１５Ｅでは、単位処理情報の整列順に、番号１、２、３、…、１２を付けてある。図中で２文字目、５文字目、１２文字目、にダミービットが付加されている。なお、辞書登録文字との比較を行う際には、ダミービット部分はマスクされる。

【００７２】

図１５Ｆ、図１５Ｇは、入力サンプルの最大語長（ビット数）$L_{S\ max}$よりも小さい単位語長N_Cを用いる場合の別の変換例である。

図１５Ｆでは、各入力サンプルｓ１，ｓ２，…をＬＳＢ（最下位ビット）側に整列し、有効桁よりもＭＳＢ側にはダミービット（例えば０）を付加して語長（ビット数）を拡張する。拡張に際しては、ビット数が入力サンプルの最大語長（ビット数）$L_{S\ max}$以上、且つ単位語長N_Cの整数倍になるようにダミービットを付与する。次に、ＬＳＢ側からＭＳＢ側に向かってマスクＬＺ処理の単位処理情報（１文字）のビット数（語長）N_Cずつ切り出して単位処理情報（１文字）に分割する。

【００７３】

図１５Ｇでは、各入力サンプルｓ１，ｓ２，…をＭＳＢ（最上位ビット）側に整列し、有効桁よりもＬＳＢ側には固定値例えば０のダミービットを付加して語長（ビット数）を拡張する。拡張に際しては、ビット数が入力サンプルの最大語長（ビット数）$L_{S\ max}$以上、且つ単位語長N_Cの整数倍になるようにダミービットを付与する。次に、ＭＳＢ側からＬＳＢ側に向かってマスクＬＺ処理の単位処理情報（１文字）のビット数（語長）N_Cずつ切り出して単位処理情報（１文字）に分割する。

【００７４】

単位処理情報変換例２（長い一定語長）

図１６Ａに、各入力サンプルが一定の大きな語長（ビット長）L_Sであり、且つ、各サンプルの有効桁数L_Eが語長L_Sに比べて短い場合のマスクＬＺ方法の単位処理情報（１文字）の変換例示す。マスクＬＺ方法の単位処理情報（１文字）のビット数N_Cを、入力サンプルの語長L_Sと同じ長さに取る。次に、図１６Ｂに示すようにサンプル内の有効なビットをＬＳＢ側に揃えて、ダミービット（例えば０）を設定する。なお、図１６Ｃに示すように、サンプルｓ１，ｓ２，…内の有効なビットをＭＳＢ側に揃えて、誤差を許容する部分にダミービット（例えば０）を設定してもよい。

【００７５】

図１６Ｄに、各入力サンプルの有効なビットをＬＳＢ側に揃え、サンプル語長L_Sよりも短いビット数の単位処理情報のビット数N_Cごとに、サンプルから有効なビットの部分を切り出す方法を示す。この方法では、まず、各入力サンプルｓ１，ｓ２，…の有効なビットをＬＳＢ側に揃える。そして、サンプル語長L_Sよりも短いビット数の単位処理情報のビット数（１文字のサイズ）N_Cごとに、サンプルｓ１から有効なビットの部分を切り出す。切り出しに際しては、切り出された有効なビットの部分のビット数が、単位処理情報のビット数N_Cより少なくなった場合は、ビット数がN_Cになるまでダミービットを加える。その後は、サンプルｓ１に対する切り出しを中止する。そして、サンプルｓ２から

、有用なビットの部分をN_Cビット数ごとに切り出す（有効桁部分より上位ビットに対し、誤差を許容することはあり得ないからである）。切り出した単位処理情報（１文字）に対する番号付けは、最初のサンプルから順に、そのＬＳＢ側からＭＳＢ側に向って行う。

【００７６】

図１６Ｅに、各入力サンプルの有効なビットをＭＳＢ側に揃え、サンプル語長L_Sよりも短いビット数の単位処理情報のビット数N_Cごとに、サンプルから有効なビットの部分を切り出す方法を示す。この方法では、まず、各サンプルの有効桁部分をＭＳＢ側に揃える。そして、最初のサンプルのＭＳＢ側からＬＳＢ側に順次単位処理のビット数（１文字サイズ）N_Cごとに切り出す。この場合は、１サンプルに対する有効桁部分の切り出しが終っても、誤差を許容する部分を全てマスクするため、その全サンプル長L_Sに対し、切り出しを行う。そして、有効桁でない部分はダミービットを付加する。ダミービットの部分にも単位処理情報（１文字）に対する番号付けを行う。図１６Ｄの例では、図１５Ｆと同様に、ダミービットを加える際に、有効桁以上の単位処理情報語長N_Cの整数倍、かつ付加するダミービットのビット数が最小となるように処理を行っている。なお、最大語を超えるまでダミービットのみの文字（単位処理情報）を追加するようにすることもできる。

【００７７】

単位処理変換構成及び処理手順

ディジタル値のサンプル列を、マスクＬＺ方法により圧縮符号化する機能構成例を図１７に、サンプル列を単位処理情報列に変換する処理手順の例を図１８に示す。

入力端子７１$_S$よりサンプル列が、入力端子７１$_M$よりサンプル毎の有効桁情報I_{ED}が、記憶部７２に入力される(ステップＳ２１)。制御部７６は、全てのサンプルに対するステップＳ２３～Ｓ３３の処理が完了したかを調べる(ステップＳ２２)。完了していなければ、片寄せ部７３は、未処理サンプル中の初めの1サンプルと、そのサンプルのビット長L_Sと、そのサンプルと対応する有効桁情報I_{ED}を、記憶部７２から取り出す（ステップＳ２３）。片寄せ部７３は、取り出されたサンプル列を、図１５、図１６で説明したようにＭＳＢ側又はＬＳＢ側に片寄せし、バッファに保持する(ステップＳ２４)。図１６に示したように、サンプルの語長L_Sが固定の場合は、有効桁部分を片寄せする。例えば、有効桁数とI_{ED}と最長(固定)ビット長L_Sとに応じて、サンプルをシフト処理することにより片寄せが行われる。

【００７８】

制御部７６は、サンプル処理長Ｌをサンプルのビット長L_Sに初期設定する(ステップＳ２５)。そして、サンプルビット長Ｌが０よりも大きいかを調べる(ステップＳ２６)。Ｌが０よりも大きければ、Ｌが単位処理ビット数(１文字サイズ)N_C以上か否かが調べられる(ステップＳ２７)。$N_C \geqq L$の場合は、図１５Ｂ、１５Ｃ、１６Ｂ及び１６Ｃに対応する処理となり、$N_C < L$の場合は図１５Ｄ、１５Ｅ、１６Ｄ及び１６Ｅに対応する処理となる。

【００７９】

ステップＳ２７で、$L \geqq N_C$と判断されると、単位ビット分離部７４は、片寄せ側（ＬＳＢ側に片寄せた場合はＬＳＢ側、ＭＳＢ側に片寄せた場合はＭＳＢ側）から単位処理情報の所定ビット数（１文字サイズ）N_Cを、ビット長Ｌのサンプルから取り出し、１文字（単位処理情報）として出力する(ステップＳ２８)。マスクデータ生成部７５は、出力された１文字に対応するマスクデータＭ１（ｉ）を生成する（ステップＳ２９）。この場合は、マスクデータ生成部７５中の生成部７５ａで各桁(ビット)が“１”のマスクデータのＭ１（ｉ）を出力する。その後、サンプルビット長Ｌから文字サイズN_Cが減算され、Ｌが更新されてステップＳ２６に戻る(ステップＳ３０)。

【００８０】

ステップＳ２７で$N_C < L$と判定されると、単位ビット分離部７４は、片寄せ側からＬビットを取り出す。ダミー付与部７４ａは、$N_C - L$ビット分（１文字サイズN_Cに対する不足分）のダミービットを、取り出されたＬビットの片寄せ側と反対側に付与し、１文

字（単位処理情報）として出力する(ステップＳ３１)。つまりサンプルをＬＳＢ側に片寄せた場合は、ダミービットは取り出されたＬビットのＭＳＢ側に付与される。マスクデータ生成部７５は、出力される１文字に対するマスクデータＭ１（ｉ）を生成する(ステップＳ３２)。この場合は、生成部７５ｂは、有効桁のＬビットに対応する各ビットは“１”、N_C－Ｌ個のダミービットに対応する各ビットは“０”のマスクデータＭ１（ｉ）が出力される。次にサンプルビット長Ｌは、０に更新されてステップＳ２６に戻る(ステップＳ３３)。

【００８１】

ステップＳ２６でサンプル処理長Ｌが０より大きくないと判定されると、ステップＳ２２に戻る。ステップＳ２２で全サンプルに対するステップＳ２３からＳ３３の処理が完了すれば、この単位処理列への変換処理は終了する。

なおステップＳ３１において、Ｌ＝N_Cの場合はN_C－Ｌ＝０となりダミービットの付与は行われない。図１７中の制御部７６は、サンプル列などの取り込み、記憶部７２から所要なサンプルの取り出しなどを行うと共に、各部を順次動作させる。制御部７６内には、その制御に必要なパラメータとしてのサンプルビット長Ｌの格納部７６ａ、有効桁情報I_{ED}の格納部７６ｂ、文字サイズN_Cの格納部７６ｃ、更にステップＳ２２、Ｓ２６、Ｓ２７における各判定を行う判定部７６ｄ、N_C－Ｌなどの演算を行う演算部７６ｅなどが設けられている。N_Cは単位ビット分離部７４内に格納してもよい。このようにしてサンプル列が単位処理情報列（文字列）に変換される。

【００８２】

この文字列と、各文字に対応するマスクデータＭ１（ｉ）の列とが、実施例１で説明したマスクＬＺ符号化部７８に入力される。そして、前述したように符号化処理がされて、符号が出力端子４４に出力される。更に必要に応じて、例えば１０２４などの所定個数のサンプルによる区間（いわゆるフレーム）ごとに、有効桁情報I_{ED}を表わす符号が、端子１７１から出力される。

【００８３】

サンプル列変換構成及び処理手順

入力符号を復号化し、その復号化した情報を、図１７の入力サンプル列に変換するための機能構成を図１９に示す。また、文字列（単位処理情報列）をサンプル列に変換する処理手順の例を図２０に示す。

入力端子５１からの入力符号が、マスクＬＺ復号化部８１に入力される。また、必要に応じて入力端子８２からのサンプルごとの有効桁情報が、マスクデータ変換部８３で文字（単位処理情報）ごとのマスクデータに変換されて、マスクＬＺ復号化部８１に入力される。端子８２からの有効桁情報は、例えばフレームごとに入力される。マスクＬＺ復号化部８１は、実施例Ａ－１～Ａ－５で述べた各種の復号化装置が用いられる。マスクデータ変換部８３の代わりに、図１０中の端子５１’の入力に相当する有効桁情報又は有効桁符号が入力されることもある。

【００８４】

マスクＬＺ復号化部８１が復号化した文字列（単位処理情報列）は、端子５６からサンプル列変換部８４に入力される（ステップＳ４１）。制御部８８は、全ての文字（単位処理情報）に対する変換処理が完了しているか確認する（ステップＳ４２）。完了していなければ、制御部８８は、復号化後の先頭サンプルの語長（ビット長）L_Sを取得する（ステップＳ４３）。単位ビット結合部８７は、内部にL_Sビットのメモリ部８７ａを確保する。また、単位ビット結合部８７は、復元ビット数L_Dを０に初期化する（ステップＳ４４）。

【００８５】

制御部８８は、L_DがL_Sより小さいか否かを調べる（ステップＳ４５）。L_Dが小さければ、単位ビット結合部８７は、未処理の文字列から初めの１文字Ｃ３（ｉ）を取り出す（ステップＳ４６）。復元ビット数L_Dと１文字のビット数N_Cが加算され、その加算結果がサンプル語長L_S以上か否かが判定される（ステップＳ４７）。L_S以上でなけれ

ば、取り出したＣ３（ｉ）がメモリ部８７ａの一端側から、書き込まれていない部分に、順次書き込まれる（ステップＳ４９）。つまり、サンプル列を単位処理情報列に変換する際に、ＬＳＢ（又はＭＳＢ）側から順次行った場合は、メモリ部８７ａへの書き込みもＬＳＢ（又はＭＳＢ）側から行い、ＬＳＢ（又はＭＳＢ）側から格納済みのL_Dビット以後にＣ３（ｉ）が格納される。

【００８６】

ステップＳ４９の後に、L_D+N_Cを新たな復元ビット数L_Dとし、ステップＳ４５に戻る（ステップＳ５０）。

ステップＳ４７の加算値がサンプル語長L_S以上であればL_S-L_Dが計算される（ステップＳ５１）。マスク文字Ｃ３（ｉ）中のＬＳＢ（又はＭＳＢ）側から書き込まれていない部分に（L_S-L_D）ビットが書き込まれてステップＳ４２に戻る（ステップＳ５２）。なおサンプルビット長L_Sを超えた$N_C-(L_S-L_D)$ビットは破棄される。破棄の方法には様々な方法がありうる。例えば、最初に$N_C-(L_S-L_D)$ビットを捨て、残ったビットを書き込んでいく方法、$N_C-(L_S-L_D)$ビットも復元用メモリに書き込み、その後でビットシフト等によって破棄する方法などである。

【００８７】

また、ステップＳ４５で復元ビット数L_Dがサンプルビット長L_Sをより小でなければ、ステップＳ４２に戻る。ステップＳ４２で全ての文字に対する処理が完了すればサンプル列への変換は終了する。

図１９中の制御部８８は、各処理に必要なパラメータL_S，N_C，L_Dなどが格納されるレジスタ８８ａ，８８ｂ，８８ｃ，x_C+L_Dの加算などを行う演算部８８ｄ，ステップＳ４２，Ｓ４５，Ｓ４８などの判定を行う判定部８８ｅなどを備え、文字列の取り込み、先頭サンプルビット長L_S、メモリ部８７ａの確保などの各種処理、各部を順次動作させる。

【００８８】

図２０のステップＳ５２の処理では、L_Sを超える$N_C-(L_S-L_D)$を破棄するが、このビットが符号化時にダミービットとして付与されたビットに対応する場合には、図１９のマスクＬＺ復号化部８１は、出力された文字列にマスク処理を行う必要が無い。すなわち図８中に破線１５７′で示したように、情報取得部５３からの出力は、マスク計算部１５２での処理を行わないで、文字列合成部５５に出力される。そして、文字列合成部５５は、符号解析部５２から入力されたマスク不一致文字と、情報取得部５３からの文字列を連結して復号文字列として出力する。また、破線１５９で示すように、この出力された文字列を辞書登録部５７で辞書に登録する。

【００８９】

このような構成とすることによって、図１９中のマスクＬＺ復号化部８１での処理を簡略化出来る。

同様に、図１０の有効桁復号化部１５６の出力を、図１９のマスクデータ生成部８５への入力とすることもできる。この場合、図１０のマスクデータ生成部１５１、マスク計算部１５２を省略できる。具体的には、図１０に破線で示すように、情報取得部５３からの出力を復号文字列として出力する。そして、有効桁情報を端子５６’に出力する。この出力が図１９の端子５６’に入力され、マスクデータ変換部８３の処理も不要となる。

【００９０】

［実施例３］

実施例３は、浮動小数点形式サンプル列の符号化、復号化、特に非特許文献４に示す符号化、復号化にこの発明を適用したものである。

符号化側

図２１にその符号化装置の例を、図２２にその処理例をそれぞれ示す。入力端子９１から、ＩＥＥＥ－７５４規格の浮動小数点形式サンプル列が入力される。整数誤差分離部９２は、各サンプルを例えば２４ビット、１６ビットなどの所定ビット数の整数信号Ｙと、その整数信号と入力サンプルとの差分信号Ｚとに分離する（ステップＳ６１）。入力サン

プル列（例えばＩＥＥＥ－７５４規格の３２ビット浮動少数点形式のオーディオ信号）は、小数点以下の端数を含む浮動小数点形式信号である。整数化部９３は、各サンプルから、極性を加味した２の補数表現の整数形式信号Ｙを生成する。差分生成部９４は、浮動小数点形式に変換した信号Ｙと入力サンプルとの差分を出力する。差分信号Ｚは、入力サンプルの仮数部の下位（２３－ｎ）ビットからなる。桁数計算部９３ａは、整数形式信号Ｙの桁数ｎを計数し、差分信号Ｚの有効桁情報として出力する（ステップＳ６２）。ここで、非特許文献４に記載されている方法を用いて、$2^{n} \leqq |Y| < 2^{(n+1)}$となるように整数形式信号Ｙの有効桁数ｎを決定する。非特許文献４では、整数形式信号のサンプルをＭとして$2^{(n-1)} \leqq |M| < 2^{(n)}$のとき仮数部の上位ｎ－１ビットが０となるとされている。これは、本発明での、$2^{n} \leqq |Y| < 2^{(n+1)}$のとき仮数部の上位ｎビットが０となることと等価である。なお、｜Ｙ｜＝０のときには、後に実施例５で示すように、入力された符号小数点信号の３２ビット全てを差分信号Ｚとしてもよい。

【００９１】

差分生成部９４からの差分信号は、入力浮動小数点形式サンプルの２３ビット仮数部の上位側の常に０となるｎビットが除かれた（２３－ｎ）ビットの可変長サンプルである。整数形式信号Ｙと差分信号Ｚとに変換した例を図２３に示す。整数形式信号Ｙは最長ビット数が例えば２４ビットであり、差分信号Ｚはビット数L_Iが最長で２３ビット、最小で１ビットの可変長信号である。

図２１の単位処理列変換部９５は、可変長の差分信号Ｚを、図１７と同じ構成、図１８と同じ処理手順で所定ビット数の文字列（単位処理情報列）に変換する（ステップＳ６３）。整数形式信号Ｙの桁数ｎから、差分信号Ｚの有効桁数（２３－ｎ）ビットが得られるから、ｎは有効桁数情報ともいえる。マスクＬＺ方法の１文字（単位処理情報）のビット数N_Cは、L_E＝２３ビットとして図１６Ｂ又は図１６Ｃに示した文字列、あるいは１文字のビット数N_Cを例えば８ビットとして図１６Ｄ又は図１６Ｅに示した変換形式としてもよい。 図１６Ｂに示した変換形式を適用すると、マスクデータＭ１（ｉ）の上位のｎビットがダミービット、すなわち有効桁ではないことを示すビットとなる。また、差分信号Ｚに対するマスクデータＭ１（ｉ）の生成は、整数桁数ｎに基づき生成することができる。差分信号Ｚは、最小でも１ビットであるため、マスクデータは最低でも１ビットの“１”をもつことになる。差分信号Ｚの有効桁の最大長は、２３ビットである。しかし、全ての差分信号Ｚで、２４ビットになるように（例えば図１６Ｅに示したように）１ビットのダミービット“０”を付加して文字列に変換すれば、文字サイズN_Cを８ビットとした場合には、差分信号ＺをN_C＝８単位で変換できる。同様に差分信号Ｚに図１５Ｅに示した変換形式を適用してもよい。

【００９２】

マスクＬＺ符号化部９６は、単位処理列変換部９５で変換した文字列を、符号化する（ステップＳ６４）。このマスクＬＺ符号化の具体的数値例として、実施例Ａ－３のマスクＬＺＷを適用した場合を説明する。辞書部４３（図５）には、８ビットで表現可能な全ての組合せをあらかじめ初期値として設定する。

辞書部４３のインデックスに用いる情報量は、初期状態では、たとえば９ビットとする。文字又は文字列が、辞書部４３にインデックスごとに登録される。９ビットで表現可能な最大文字列の数に達した場合には、インデックスの表現に用いるビット数を１ビットずつ増加させていくようにする。インデックスに用いるビット数の最大は、たとえば１６ビットとする。

【００９３】

入力文字列Ｃ（ｉ＋ｋ）および有効桁情報（マスクデータ）Ｍ（ｉ＋ｋ）を順に入力し、辞書部４３に登録された文字列の中で最長の一致文字列を探索する。この探索は、図７中のステップＲ１０～Ｒ６で行う。符号生成部４９は、探索によって得られた最長に一致する文字列の辞書インデックスj_{max}を、符号として出力する。入力された文字をＣ１（ｉ＋ｋ）、その有効桁情報（マスクデータ）をＭ１（ｉ＋ｋ）とし（ｋを０から次第に増加させる）、辞書部４３に登録されたｊ番目（インデックス）の文字列の長さをK_jとし

、その文字列をＤ（ｊ）（ｋ）［ｋ：０≦ｋ＜K_j］で表わすとする。この場合、一致判定は、ｋが０からK_j－１までの全ての（ｋ）について、Ｃ１（ｉ＋ｋ）＆Ｍ１（ｉ＋ｋ）＝Ｄ（ｊ）（ｋ）＆Ｍ１（ｉ＋ｋ）が成り立つような、最長のｋを有する辞書部インデックスj_{max}を求める処理である。

【００９４】

符号を出力するごとに、直前に出力した符号と対応する文字列Ｄ（j_{max}’）（ｋ）［ｋ：０≦ｋ＜k_{max}’］に、今回出力した符号と対応する文字列の先頭の一文字Ｄ（j_{max}）（０）をＭ１（ｉ＋０）でマスク処理したマスク先頭文字Ｄ（j_{max}）（０）＆Ｍ１（ｉ＋０）を加えた文字列を新たに辞書登録する。このようにして次第に長い文字列が辞書部に登録され、より長い文字列との一致を探索することが可能となる。

整数形式信号Ｙは、整数符号化部９７で圧縮符号化される（ステップＳ６４）。この圧縮符号化は、整数値としての波形値の相関などを利用して、可逆圧縮法により効率よく圧縮符号化することができる。合成部９８は、整数形式信号の符号C_Yと、マスクＬＺ符号化部９６からの差分信号Ｚに対する符号化符号C_Zとを合成する。そして、合成部９８は、合成された符号を、サンプルごとあるいはフレームごとに、符号化符号として出力する（ステップＳ６５）。

【００９５】

復号化側

図２１に対応する復号化装置の機能構成を図２４に、その処理手順の例を図２５にそれぞれ示す。分離部１０２は、入力端子１０１からの符号C_Y及び符号C_Zの組を、符号C_Yと符号C_Zに分離する（ステップＳ７１）。整数復号化部１０３は、符号C_Yを、整数形式信号Ｙに復号する（ステップＳ７２）。浮動小数点化部１０４は、整数形式信号Ｙを、浮動小数点形式信号に変換する（ステップＳ７３）。

【００９６】

符号C_Zは、マスクＬＺ復号化部１０５に入力される。桁計算部１０３ａは、整数復号化部１０３で復号された整数形式信号Ｙの桁数ｎを計数し、桁数ｎを有効桁情報としてマスクＬＺ復号化部１０５に出力する（ステップＳ７４）。また、点線で示したように、マスクデータ変換部１０６を整数復号化部１０３とマスクＬＺ復号化部１０５との間に備えても良い。この場合は、有効桁情報ｎはマスクデータ変換部１０６に入力される。マスクデータ変換部１０６は、（２３－ｎ）ビットを各サンプルの有効桁とし、文字（単位処理情報）ごとのマスクデータを生成して、マスクＬＺ復号化部１０５に出力する。ここで、整数形式信号Ｙの有効桁数ｎは、符号化時の処理と同じ方法で決定される。すなわち、$2^n \leqq |Y| < 2^{(n+1)}$となるようにｎを決定する。

【００９７】

マスクＬＺ復号化部１０５は、図１０及び図１１に示した構成及び処理を行う復号化装置と同じである。マスクＬＺ復号化部１０５は、符号C_Zを、単位処理情報列にマスクＬＺ復号化する（ステップＳ７５）。マスクＬＺ復号化部１０５で復号化された文字（単位処理情報）列は、サンプル列変換部１０７に入力される。サンプル列変換部１０７は、図１９のサンプル列変換部８４と同様の構成である。サンプル列変換部１０７は、マスクＬＺ復号化部１０５で復号化された文字列を、有効桁情報を用いてマスク処理する。マスク処理の後、ダミービットを除去して文字列を差分信号Ｚに変換する（ステップＳ７６）。この差分信号Ｚは各サンプルとも０以外になり得る（２３－ｎ）ビットで構成される。組立部１０７ａは、差分信号Ｚを、浮動小数点の２３ビット仮数部信号に変換する（ステップＳ７７）。合成部１０８は、仮数部とされた差分符号Ｚと、復号された整数形式信号Ｙの浮動小数点信号とを合成して、原浮動小数点信号サンプルを出力する（ステップＳ７８）。

【００９８】

本実施例では、差分信号Ｚとして０以外の値となり得る可変長ビット列に対し、マスクＬＺ符号化、復号化を行った。しかし、２３ビットの仮数部のままの差分信号Ｚに対し、マスクＬＺ符号化、復号化を行ってもよい。この場合は、サンプル長は固定語長となる。

【００９９】
［実施例４］
例えば、ユニバーサル符号化方法で符号化する場合に、入力サンプル列を共通の乗数で除算した結果を符号化すると圧縮率を高めることができる。本実施例は、この共通乗数で除算して符号化する方法に、この発明を適用したものである。共通乗数の除算を行うと圧縮率を向上できることをまず簡単に説明する。
入力信号サンプル列がｘ（１），ｘ（２），ｘ（３）であり、図２６Ａに示すように１０進数表現でｘ（１）＝２５０、ｘ（２）＝５０、ｘ（３）＝３５０の場合、その２進数表現はそれぞれ、「０」と「１」が比較的ランダムに配列されている。
【０１００】
サンプルｘ（１），ｘ（２），ｘ（３）を共通の数Ａ＝１．５６２５で割算すると、その商信号ｙ（１），ｙ（２），ｙ（３）は、図２６Ｂに示すようにそれぞれ１０進数表現でｙ（１）＝１６０、ｙ（２）＝３２、ｙ（３）＝２２４となる。
商信号ｙ（１），ｙ（２），ｙ（３）の２進数表現は、それぞれ図２６Ｃに示すように、「０」が連続して存在する部分が多い配列となる。従って、ｙ（１），ｙ（２），ｙ（３）を、高い圧縮率で符号化できる。ただし、乗算Ａも送出する必要がある。しかし、サンプル数が多く、かつ、大部分が高圧縮が可能な商信号にできる場合は、全体としての圧縮符号量を大きく減少させることができる。
【０１０１】
図２７に、ＩＥＥＥ－７５４浮動小数点数のサンプル列に適用した例を示す。１０進数表現の数値４７８．４，９５．６８，６６９．７６の浮動小数点表現は、図２７Ａに示すように、仮数部の２３ビットの「１」、「０」の配列は、どれもかなりランダムである。しかし、これらを共通乗数２．９９で割算すると、その商信号は、図２７Ｂに示すように、１０進数表現が１６０，３２，２２４にそれぞれなる。浮動小数点表現での仮数部の２３ビットの「１」、「０」の配列では、「０」が非常に多くなる。従って、図２７Ａのまま圧縮符号化するよりも、著しく高い圧縮率で符号化することができる。
【０１０２】
符号化側
図２８を参照して実施例４の符号化側を説明する。
入力端子２０１からの入力信号Ｘ＝（ｘ（１），ｘ（２），…）はディジタル化されたサンプル列であり、これが入力されると（ステップＳ８１）、区間（フレーム）分割部２０２で所定数Ｎ個（例えば１０２４個）のサンプル列に分割され、図に示していないが一旦記憶部に記憶される（ステップＳ８２）。サンプル列としては、音響信号の場合は、２４ビットの量子化ビット数で量子化された整数サンプル列や３２ビット単精度浮動小数点形式サンプル列などが考えられる。カラー画像信号の場合は各色情報要素に分解してラスタ走査したピクセル情報のディジタル化されたサンプル列である。
【０１０３】
分割区間ごとの入力信号としてのサンプルｘ（ｉ）（ｉ＝０，１，…，Ｎ－１）の集合は、１フレームごとに乗数推定部２０３に渡される。乗数推定部２０３は、全てのサンプルｘ（ｉ）（ｉ＝０，１，…，Ｎ）に対して、共通の乗数Ａを推定する（ステップＳ８３）。例えば、全てのサンプルｘ（ｉ）が共通の値９９．０で割り切れる場合には、共通の乗数Ａ＝９９．０とする。この共通の乗数Ａを推定する方法としては、複数種類の方法が考えられる。例えば、近似共通ファクタ（ＡＣＦ：Approximate Common Factor）の有理近似を用いる。ここでは、適切な乗数Ａが与えられるものとする。
【０１０４】
乗数推定部２０３で決定された乗数Ａは、除算処理部２０４、乗算部２０５、乗数符号化部２０６に渡される。
除算処理部２０４では、乗数推定部２０３より渡された乗数Ａと、Ｎ個のサンプルｘ（ｉ）を入力とし、Ｎ個の商信号ｙ（ｉ）＝ｘ（ｉ）／Ａを算出する（ステップＳ８４）。このとき、ｙ（ｉ）は整数形式、浮動小数点形式、固定小数点形式のいずれでもよい。そ

の決められた表現形式に変換するときに、切り捨、切り上げ、四捨五入、nearest tie to even（ニアレストタイツウイーブン）などの丸め処理を行ってもよい。

【０１０５】

例えば、ｘ（ｉ）を倍精度の浮動小数点数、ここでは６４ビットの浮動小数点数に変換して乗数Ａで除算する。得られた倍精度の商を、最も近い単精度（３２ビット）の浮動小数点数に丸めて商信号ｙ（ｉ）とする。

除算処理部２０４で得られたＮ個の商信号列Ｙ＝（ｙ（０），ｙ（１），…，ｙ（Ｎ－１））は、商信号符号化部２０７と、この例では乗算部２０５に渡される。

乗算部２０５では、乗数推定部２０３から渡された乗数Ａを、除算処理部２０４から渡されたＮ個の商信号列Ｙの各信号ｙ（ｉ）にそれぞれ乗算して、復元されたＮ個のサンプルｘ’（０），ｘ’（１），…，ｘ’（Ｎ－１）を得る（ステップＳ８５）。

【０１０６】

このとき、復元サンプルｘ’（ｉ）は３２ビットの浮動小数点表現で表現可能な値の範囲に丸められる。例えば、ｙ（ｉ）とＡを掛け合わせた結果を倍精度（６４ビット）の浮動小数点数として保持し、得られた倍精度の乗算結果を、最も近い単精度（３２ビット）の浮動小数点数に丸めてｘ’（ｉ）とする。得られたサンプル列Ｘ’＝（ｘ’（０），ｘ’（１），…，ｘ’（Ｎ－１））は、誤差算出部２０８に渡される。

誤差算出部２０８は、入力信号から取り出したＮ個のサンプルｘ（０），ｘ（１），…，ｘ（Ｎ－１）それぞれから乗算部２０５から渡されたＮ個の復元サンプルｘ’（０），ｘ’（１），…，ｘ’（Ｎ－１）を差し引いて、Ｎ個の差分信号ｚ（０），ｚ（１），…，ｚ（Ｎ－１）から成る差分信号列Ｚ＝（ｚ（０），ｚ（１），…，ｚ（Ｎ－１））を得る（ステップＳ８６）。この差分信号の算出は減算を用いる代わりに、ｘ（ｉ）とｘ’（ｉ）の３２ビットをそのままビット単位の排他的論理和演算（ｘｏｒ）を行ってもよい。要するにｘ（ｉ）とｘ’（ｉ）との差分演算を行えばよい。

【０１０７】

この誤差算出部２０８からの差分信号列Ｚは、有効桁生成部２０９及び単位処理列変換部２１１に入力される。有効桁生成部２０９は、入力されたＮ個の差分信号ｚ（０），ｚ（１），…，ｚ（Ｎ－１）中の最も上位側に“１”があるビット位置（桁）を求める。つまりフレーム内の最大ビット長を、有効桁情報として求める（ステップＳ８７）。そして、この有効桁情報が単位処理列変換部２１１に入力される。単位処理列変換部２１１は、例えば図１７に示した機能構成と同じであり、図１８に示した処理手順により、差分信号列を、所定ビット数の文字列（単位処理情報列）に変換する。そして、単位処理列変換部２１１は、図１５Ｂ～図１５Ｇ、図１６Ｂ～図１６Ｅのいずれかに示す文字列（単位処理情報列）に変換された文字列と有効桁情報を、マスクＬＺ符号化部９６に出力する（ステップＳ８８）。

【０１０８】

商信号符号化部２０７は、除算処理部２０４からの商信号列Ｙを、圧縮符号化して、商符号C_Yを出力する。圧縮符号化の方法としては、線形予測符号化方法などの波形値の相関を利用した高圧縮率の可逆圧縮符号化方法（例えば非特許文献３参照）やその他の非可逆の圧縮符号化法（ＭＰＥＧ４、ＡＡＣ、TwinＶＱなど）がある。マスクＬＺ符号化部９６は、単位処理列変換部２１１からの文字列を圧縮符号化し、単位処理符号C_Zとして出力する。このマスクＬＺ符号化部９６では、先に述べたマスクＬＺ７７、マスクＬＺ７８、マスクＬＺＷなどのマスクＬＺ符号化が行われる。有効桁符号化部２１２は、フレーム内の最長ビット長を表わす有効桁情報を、有効桁符号C_dに可逆符号化する。乗数符号化部２０６は、乗数Ａを可逆符号化し、乗数符号C_Aとして出力する（ステップＳ８９）。合成部９８は、商信号符号C_Y、単位処理符号C_Z、有効桁符号C_d、乗数符号C_Aをフレームごとに１組の符号C_xとして出力する（ステップＳ９０）。商信号の符号化は、ステップＳ８４の後であればいつでもよく、乗数Ａの符号化はステップＳ８３の後であればいつでもよく、有効桁情報の符号化もステップＳ８７の後であればよい。

【０１０９】

復号化側

図３０にこの実施例４の復号化装置の機能構成例を、図３１にその処理手順の例を示す。

分離部１０２は、端子２２１からの符号化データC_xを、商信号符号C_Y、単位処理符号C_Z、有効桁符号C_d、乗数符号C_Aに分離する（ステップＳ９１）。商信号復号化部２２２は、商信号符号C_Yを、商信号符号化部２０７の符号化方法と対応する復号化方法によりＮ個の商信号ｙ（０），ｙ（１），…，ｙ（Ｎ－１）に復号する（ステップＳ９２）。乗数復号化部２２３は、乗数符号C_Aを復号する（ステップＳ９３）。乗算部２２４は、復号化された乗数Ａを復号されたＮ個の商信号ｙ（０），ｙ（１），…，ｙ（Ｎ－１）にそれぞれ掛算する（ステップＳ９４）。有効桁復号化部２２５は、有効桁符号C_dを可逆復号し、有効桁情報（フレーム内で最長のビット長を示す情報）を生成する（ステップＳ９５）。マスクＬＺ復号化部１０５は、前記の有効桁情報を用いて、単位処理符号C_Zを文字列に復号化する（ステップＳ９６）。サンプル列変換部２１１は、マスクＬＺ復号化部１０５で復号化された文字列を、サンプル列の差分信号ｚ（０），ｚ（１），…，ｚ（Ｎ－１）に変換する（ステップＳ９７）。加算部２２６は、差分信号ｚ（０），ｚ（１），…，ｚ（Ｎ－１）を、乗算部２２４の出力に対してそれぞれ加算する（ステップＳ９８）。フレーム連結部２２７は、加算部２２６からの出力信号ｘ（０），ｘ（１），…，ｘ（Ｎ－１）を連結し、復号信号として出力する（ステップＳ９９）。ｙ（ｉ）とＡとの掛算信号Ａｙ（ｉ）を求める処理と、差分信号サンプルｚ（ｉ）を求める処理はいずれを先に行ってもよい。有効桁符号C_dや乗数符号C_Aは、前フレームと同一の場合は、そのことを示す少ないビット数の符号を出力することにしてもよい。商信号符号化部２０７として非可逆圧縮符号化が使用される場合は、図２８中に破線で示すように、商信号符号C_Yを復号化部２１３で復号し、その復号信号を除算処理部２０４の出力信号ｙ（ｉ）の代わりに乗算部２０５に入力すればよい。

【０１１０】

［実施例５］

共通乗数で除算して圧縮符号化する方法と、非特許文献４に示す浮動小数点信号の符号化方法に、この発明の方法を適用した実施例５を説明する。

符号化側

図３２に機能構成例を、図３３に処理手順の例をそれぞれ示す。

実施例４と同じように、区間分割部２０２は、入力信号ｘ（ｉ）を取込み（ステップＳ８１）、フレーム分割して記憶部に記憶する（ステップＳ８２）。乗数推定部２０３は、乗数Ａを推定する（ステップＳ８３）。除算処理部２０４は、乗数Ａで入力信号ｘ（ｉ）を除算処理し、有効桁以内に丸める（ステップＳ８４）。なお、乗数Ａが1のとき、除算処理部２０４での除算処理を省略してもよい。この例では、判定処理部２３１内の判定部２３１ａは、除算処理結果の単精度浮動小数点数が、無限大の値や、非正規化数、数値として表すことができないＮａＮという特殊数値であるか否かを判定する（ステップＳ１１０）。以下、無限大の値、非正規化数、数値として表すことができないＮａＮを総称して「特殊数値」いう。ステップＳ１１０の判定が特殊数値であれば、スイッチ２３１ｂが除算処理部２０４側から０信号源２３１ｃ側に切替えられ、０信号が整数化部９３へ供給される（ステップＳ１１１）。ステップＳ１１０の判定が特殊数値でなければ、ステップＳ８４の除算処理結果が整数化部９３へ入力される。

【０１１１】

整数化部９３は、判定処理部２３１からの入力信号から整数形式信号ｙ（ｉ）を生成する（ステップＳ１１２）。浮動小数点化部２３２は、整数形式信号ｙ（ｉ）を浮動小数点信号に変換する（ステップＳ１１３）。乗数部２０５は、この浮動小数点信号に乗数Ａを乗算し、掛算結果を丸め処理する（ステップＳ１１４）。ここで、特殊数値であると判定される場合や、ｙ（ｉ）＝０で乗算結果が０となる場合には、入力信号（サンプル）ｘ（ｉ）の３２ビットが誤差算出部２０８をそのまま通過して 単位処理列変換部９５に入力される。なお、乗数Ａが1のとき、ステップＳ１１４の乗算処理を省略してもよい。また

、ステップＳ１１３とＳ１１４を破線で囲んだように、ステップＳ１１３とステップＳ１１４とを、整数信号ｙ（ｉ）に浮動小数点の乗数Ａを乗算することで一度に行ってもよい。

【０１１２】

一方、乗数判定部２３３は、乗数Ａが１であるか否かを判定する（ステップＳ１１５）。Ａ＝１であれば、スイッチ２３４は、差分生成部９４に接続され、入力信号ｘ（ｉ）が差分生成部９４に入力される。整数化部９３の桁数計算部９３ａは、整数形式信号ｙ（ｉ）の桁数ｎを計数し、有効桁情報を生成する。この有効桁情報ｎも差分生成部９４に入力される（ステップＳ１１６）。差分生成部９４は、実施例３の符号化装置と同じように、入力信号（サンプル）ｘ（ｉ）の仮数部の下位（２３－ｎ）ビットからなる差分信号z_e（ｉ）を生成する（ステップＳ１１７）。なお、差分信号z_e（ｉ）は可変長サンプルである。ここで、ｘ（ｉ）が特殊数値の場合や、ｙ（ｉ）＝０となる場合には、入力信号（サンプル）ｘ（ｉ）の３２ビットが単位処理列変換部９５に入力される。ステップＳ１１５で、乗数Ａが１でなければ、スイッチ２３４は誤差算出部２０８に接続される。誤差算出部２０８は、入力信号ｘ（ｉ）と乗算部２０５の出力信号との差分信号z_d（ｉ）を生成する（ステップＳ１１８）。

【０１１３】

単位処理列変換部９５は、誤差算出部２０８からの差分信号z_d（ｉ）または差分生成部９４からの差分信号z_e（ｉ）を受け取り、単位処理情報列に変換する（ステップＳ６３）。整数符号化部９７は、整数形式信号ｙ（ｉ）を整数符号C_Yに符号化する。マスクＬＺ符号化部９６は、単位処理情報列を単位処理符号C_ZにマスクＬＺ符号化する。乗数符号化部２０６は、乗数Ａを乗数符号C_Aに符号化する（ステップＳ１１９）。合成部９８は、符号C_Y，C_Z，C_Aを合成し、フレームごとに出力する（ステップＳ６５）。

【０１１４】

判定部２３１ａでｘ（ｉ）が特殊数値であると判定されると、ｙ（ｉ）＝０とされる。したがって、その入力信号（サンプル）ｘ（ｉ）の３２ビットは、誤差算出部２０８を通過して単位処理列変換部９５に入力される。乗数判定部２３３でＡ＝１でないと判定されると、誤差算出部２０８からその入力信号（サンプル）ｘ（ｉ）の仮数部の２３ビットが単位処理列変換部９５に入力され、Ａ＝１と判定されると、差分生成部９４からその入力信号（サンプル）ｘ（ｉ）の仮数部中の下位（２３－ｎ）ビットが単位処理列変換部９５に入力される。単位処理列変換部９５は、図１５Ａ～図１５Ｇで説明したように、このような可変長サンプルを単位処理情報列に変換することができる。また最大サンプル長が３２ビットとなる場合も、同様に単位処理情報列に変換することができる。

【０１１５】

ここで、図３５のように、単位処理列変換部９５の直前に、最上位ビット桁判定部２３５を用いることとしても良い。最上位ビット桁判定部２３５は、当該処理区間の入力サンプル列（差分信号z_d（ｉ）もしくは差分生成部９４よりの差分信号z_e（ｉ））の０でない最上位ビット桁を判定し、単位処理列変換部９５、および、最上位桁符号化部２３７に最上位桁（ビット）情報$L_{S\ max}$として出力する。最上位桁符号化部２３７では、０でない最上位桁（ビット）の情報を符号化して合成部９８に出力する。

単位処理列変換部９５は、信号の最上位桁（ビット）情報$L_{S\ max}$を用いて、入力された信号（差分信号z_d（ｉ）もしくは差分生成部９４よりの差分信号z_e（ｉ））を、単位処理列に変換する。

【０１１６】

また、処理区間内で、整数形式信号y(i)の値が０のサンプルと０でないサンプルを分け、それぞれを連続してマスクＬＺ符号化部に入力されるようにサンプルの順序を整列するようにしても良い。例えば図３５Ｂに示すように、処理する方法がある。差分生成部９４もしくは誤差計算部２０８からの処理区間（フレーム）の最初の入力サンプルを選択する（ステップＳ１３１）。そのサンプルが、ｙ（ｉ）が０となるサンプルか、それ以外のサンプルかを判定する（ステップＳ１３２）。ここで、ｙ（ｉ）が０となるサンプルとは、

符号化するビット有効桁が３２ビットとなるサンプルに相当する。また、それ以外のサンプルとは、符号化するビット有効桁が３２ビット未満のサンプルに相当する。ステップＳ１３２がＹｅｓであれば、単位処理情報列に変換する（ステップＳ１３３）。次に、処理区間の最終サンプルに対するステップＳ１３２とＳ１３３を行ったかを判定する（ステップＳ１３４）。最終サンプルに対する処理が終わっていなければ、次のサンプルを選択してステップＳ１３２に戻る（ステップＳ１３５）。ステップＳ１３２で、ｙ（ｉ）が０でないサンプルと判定されると、ステップＳ１３４に移る。ステップＳ１３４で、最終サンプルに対する処理が終了したと判定されると、その区間の未処理サンプル、つまりｙ（ｉ）が０とならない差分信号をまとめて単位処理情報列に変換する（ステップＳ１３６）。

【０１１７】

最上位桁（ビット）情報は、符号化するビット有効桁が３２ビットのサンプル集合と、３２ビット未満のサンプル集合に対してそれぞれ判定するようにしてもよい。

復号化側

図３４にこの実施例５の機能構成例を示す。その処理手順は図３１に示したものとほぼ同様である。ただし、商信号符号C_Yは整数符号C_Yと対応している。

分離部１０２は、入力された符号C_Xを整数符号C_Y、単位処理符号C_Z、乗数符号C_Aに分離する。整数復号化部１０３は、整数符号C_Yを整数形式信号ｙ（ｉ）に復号化する。整数復号化部１０３から、整数形式信号ｙ（ｉ）の桁数ｎ及びｙ（ｉ）＝０の場合にはそのことを示す情報が、有効桁情報として、マスクデータ変換部１０６に入力される。マスクデータ変換部１０６は、有効桁情報を、単位処理情報ごとのマスクデータを変換する。つまりＡ＝１の場合、各サンプルについて（２３－ｎ）ビットと対応するマスクデータが、またｙ（ｉ）＝０の場合はそのサンプルについて３２ビットと対応するマスクデータがマスクＬＺ復号化部１０５に入力される。マスクＬＺ復号化部１０５は、単位処理情報C_ZをマスクＬＺ復号化して、単位処理情報列を生成する。乗数復号化部２２３は、乗数符号C_Aを乗数Ａに復号化する。乗数Ａが１でない場合は、マスクデータ生成部２３６は、仮数部の２３ビットと対応するマスクデータを生成し、マスクＬＺ復号化部１０５に入力する。

【０１１８】

浮動小数点化部１０４は、復号化された整数形式信号ｙ（ｉ）を浮動小数点信号に変換する。乗算部２２４は、浮動小数点化部１０４の出力に乗数Ａを乗算する。サンプル列変換部１０７は、マスクＬＺ復号化部１０５からの単位処理情報列を、サンプル列つまり差分信号ｚ（ｉ）に変換する。加算部２２６は、差分信号ｚ（ｉ）を乗算部２２４の出力に加算し、その出力がフレーム連結部２２７に入力される。なお、（２３－ｎ）ビットのサンプルの場合、サンプル列変換部１０７では、図２４中の対応する部分と同様に、その上位にｎ個の０が連結されて２３ビットの仮数部に組立てられる。復号化整数形式信号ｙ（ｉ）が０の場合は、サンプル列変換部１０７から３２ビットの浮動小数点信号（サンプル）が出力され、これがフレーム連結部２２７に入力されることになる。

【０１１９】

なお、図３２に破線で示すように、判定処理部２３１を区間分割部２０２の直後に挿入してもよい。特殊数値は例外的なものであり、特殊数値を含まない浮動小数点信号に対しては、図３２中の破線２３８のように、除算処理部２０４の出力側を整数化部９３に直接接続してもよい。

［変形実施例］

ここで、マスクＬＺ符号化部の符号化結果のサイズが、単位処理列変換部９５に入力されたサイズよりも大きい場合には、マスクＬＺ符号化を行わないことを示す情報とともに元のデータをそのまま合成部９８に出力することも出来る。

【０１２０】

この場合には、区間分割部で分割された各区間に対してマスクＬＺを用いた場合と、用いない場合のデータのサイズを比較し、サイズが小さくなるほうを選択符号と共に出力する。

各区間の符号化を開始する際に、マスクＬＺ符号化部内の内部状態を保存しておく。圧縮後にサイズが小さくならない場合には、次の区間のマスクＬＺ符号化処理を行う前にマスクＬＺ符号化部の内部状態を保存しておいた状態に復帰する。これによって符号化側と復号化側の内部状態の同期を保つことが出来る。

【０１２１】

次に、サンプルごとに有効桁が異なる入力信号の符号化に適用した実施例を説明する。

たとえば、音響信号の符号化では、聴覚心理モデル等を用いて、人間の聴覚特性から歪の知覚されにくい部分には大きな歪を許すような符号化方式がある。この許容される歪の量をもとに、値の有効桁情報が与えられるものとする。たとえば、図１６Ａに示したように、入力されるサンプル列の語長は一定で、たとえば３１ビットとする。図１７中に示した単位処理変換部７９で、マスクＬＺ符号化部７８での処理に適した形になるように語長割当処理し、有効桁情報を元にマスクデータを設定する。

【０１２２】

ここでは、図１６Ｅに示した文字列変換を例にとり、１文字の語長を８ビットとした場合とする。図１６Ｅに示したように、入力された信号列は、有効桁の情報を元に、ＬＳＢ側に揃えられ、マスクＬＺ方法の処理単位である８ビットごとに切り取って文字列として並べられる。この際、入力信号列の語長が１文字の語長の整数倍（今の例では８の倍数）になっていない場合は、８ビットの倍数になるように、ＭＳＢ側に値“０”のダミービットが追加される。

また、有効桁を表すマスクデータが作成される。マスクデータは、有効ビット部は“１”、歪を許すビット部およびダミービット部は“０”となるように設定する。入力信号から切り出されたｉ番目の文字をＣ１（ｉ）、対応するマスクデータをＭ１（ｉ）とすれば、この文字列、マスクデータ列を入力として図５に示した実施例１（実施例１の変形を含む）の圧縮符号化方法を適用することが可能となる。

【０１２３】

以下、実施例Ａ－１（マスク－ＬＺ７７）を適用した場合の具体例を示す。

先読みを２５６文字（８ビット）、辞書部４３（図１４Ａ）のバッファサイズを６５３６文字（１６ビット）とする。

これによって位置情報（インデックス）は、１６ビット、最大一致長は８ビットで表されることとなる。なお、これらは一例であり、既知の方法を用いてさらに効率的に処理することも可能である。たとえば、この発明の有効性を損なうことなくＬＺ７７のかわりにＬＺＳＳを適用することもできる。

【０１２４】

先読みバッファ６５（図１４Ａ）中の入力文字列Ｃ１（ｉ）、および有効桁情報（マスクデータ）Ｍ１（ｉ）と、辞書部４３（スライドバッファ）に登録された文字列の中で、最長の一致文字列を探索する。探索によって得られた辞書中の最長一致文字列に対応するインデックスと、一致長、一致しなかった先読みバッファ中の次の１文字$C1_D$とを、符号として出力する。

先読みバッファ中の文字列をＣ１（ｉ＋ｋ）、有効桁情報をＭ１（ｉ＋ｋ）［ｋ＝０，１，２，・・・］とし、辞書部（スライドバッファ）に登録されたｊ番目の位置から始まる文字列をＤ（ｊ）（ｋ）、有効桁をＭ（ｊ）（ｋ）で表し、辞書のｊ番目の位置から始まる文字列と、先読みバッファ内の文字列の一致長をk_{max}とすれば、一致判定では、ｋ＝０から$k=k_{max}-1$までの全てのｋについて、Ｃ１（ｉ＋ｋ）＆Ｍ１（ｉ＋ｋ）＝Ｄ（ｊ）（ｋ）＆Ｍ（ｊ）（ｋ）が成り立つような、最長のk_{max}を有する辞書中の位置ｊを求めることになる。

【０１２５】

実施例Ａ－１の代わりに、実施例Ａ－４を用いた場合には、ｋ＝０から$k=k_{max}-1$までの全てのｋについて、Ｃ１（ｉ＋ｋ）＆Ｍ１（ｉ＋ｋ）＆Ｍ（ｊ）（ｋ）＝Ｄ（ｊ）（ｋ）＆Ｍ（ｊ）（ｋ）＆Ｍ１（ｉ＋ｋ）が成り立つような、最長のk_{max}を有する辞書中の要素ｊを求めることとなる。

次に、符号を出力するごとに、一致した文字列と続く不一致１文字を先読みバッファから削除し、一致文字と続く不一致１文字を辞書部４３に登録する。辞書部４３中の最も古い文字列は破棄される。

【０１２６】

実施例Ａ－１において、一致した文字列Ｃ１（ｉ＋ｋ）とＭ１（ｉ＋ｋ）［ｋ：０≦ｋ＜k_{max}］を辞書部４３（スライドバッファ）に登録する場合に、非可逆な例では、一致した先読みバッファにあるk_{max}文字のＣ１（ｉ＋ｋ），Ｍ１（ｉ＋ｋ）をコピーするのではなく、一致して符号として出力したＤ（j_{max}）（ｋ）と、Ｃ１（ｉ＋ｋ）の有効桁情報Ｍ１（ｉ＋ｋ）とを組にして辞書部４３に追加する。次いで、最長一致した文字列の次の１文字Ｃ１（ｉ＋ｋ）（＝$C1_D$）とＭ１（ｉ＋ｋ）（＝$M1_D$）を先読みバッファ６５に追加する。つまり図１４Ａ中に破線で示すように、辞書部４３にマスクデータＭ（ｉ）が格納されるマスク部４３ｃも設けられる。

【０１２７】

先読みバッファ６５には、削除されたのと同数の新たな入力文字列と、対応する有効桁情報が読み込まれる。

復号化処理を以下に示す。

復号化処理は、符号化処理で行ったそれぞれの処理について、対応する逆の操作を行う。

図８で既に説明したように、復号化装置では符号列と、有効桁情報を入力として受取る。有効桁情報は、マスクデータ生成部１５１によってマスクデータ（Ｍ１（ｉ）列）に変換される。

【０１２８】

こうして得られた符号と、有効桁マスクデータが、マスクＬＺ復号化装置に入力される。

マスクＬＺ復号化装置は、符号入力と、有効桁情報（マスクデータ）を用いて元の文字列を復元する。こうして得られた文字列は図１９中のサンプル列変換部８４により元の語長３１ビットの信号列に復元される。

上述では入力文字（単位処理情報）は数値情報を前提とした。従って有効桁は一般に上位桁程重要なものである。有効でない、あるいは無視してもよい桁（ビット）はＬＳＢ側である。またダミービットも、ＬＳＢ側に片寄せした場合も、そのサンプルの有効桁の最上位ビット以上で、１文字サイズを満たすに必要な連続したものとなった。一方、例えば一般の文字においても、その構成ビットのある特定ビットの１～複数が変化しても情報列の全体には、ほとんど影響を与えない場合がある。この場合は、有効桁やダミービットは連続しない。１文字のビット列の途中のビット（桁）を有効でない桁としてこの発明を適用することができる。この場合、文字列をサンプル列に変換する際に、図１９中に破線で示すように文字ごとの有効桁情報から、マスクデータ生成部８５でマスクデータを生成する。そして、選択部８６は、単位ビット結合部８７に入力する文字列中の各文字から、対応するマスクデータの“１”の桁（ビット）のビットのみを選択すればよい。有効桁情報はマスクＬＺ復号化部８１又は外部から与えられる。

【０１２９】

ハッシュを用いた高速な探索処理

図６のステップＳ４の辞書探索処理にハッシュを用いて高速化する具体例を説明する。図７に示した方法では、辞書インデックスｊを０から最大値まで変化させ、辞書に登録された全ての文字列を入力文字列と比較する。マスク処理をしない従来のＬＺ符号化方法では、ハッシュ関数処理を用いて探索を高速化する方法が知られている。しかし、マスクＬＺでは、同一の辞書登録文字が、複数の入力文字と一致するかどうかを、マスクに依存して探索する必要があるため、既知のハッシュ関数処理手法をそのまま適用することは出来ない。

【０１３０】

ここでは、辞書部に文字列を登録する際に、有効桁を考慮したハッシュ関数処理を用い

て探索を高速化する方法をマスクＬＺＷに適用した例を示す。
入力文字列が辞書部に登録された内容と一致した文字列を、Ｄ（ｊ）（ｋ）[ｋ：０≦ｋ＜K_j]とし、続く不一致文字Ｃ１（ｉ＋K_j）を連結した文字列を新たに辞書部のインデックスＪ＋１に登録する場合を考える。既知のハッシュ関数処理法では、例えばｊとＣ１（ｉ＋K_j）とをキーとしてハッシュ値を算出し、辞書部内のハッシュテーブルの対応する場所にＪ登録する。

【０１３１】
これに対して、この発明では、例えば図３６に示すように辞書部３６１内にハッシュテーブル３６２を互いに異なるマスクの数だけ用意する。そして、ハッシュ関数演算部３６３は、ｊとＣ１（ｉ＋K_j）＆Ｍとをキーとするハッシュ値を算出する。切替部３６４でマスクデータＭに対応したハッシュテーブルに切替え、文字列を登録する。ここで、Ｍはマスクビットが互いに異なるマスクデータで、１文字の処理単位が８ビットの場合で、ＬＳＢ側に揃えて探索する場合にはＭは００００００００１から１１１１１１１１までの８種類である。

【０１３２】
探索を行う場合には、Ｍ（ｉ＋ｋ）で１が立っている数をキーとしてハッシュテーブルを１つ選択し、選択されたハッシュテーブル３６２をｊとＣ１（ｉ＋K_j）＆Ｍ（ｉ＋ｋ）とをキーとして探索する。
有効桁情報としては、例えば有効桁の最上位ビットと最下位ビットの２つの位置情報を用いてもよい。この場合は例えば比較すべき２つの文字に対し、その一方のビット位置と対応する各ビットが一致するかを調べ、一致すれば次の各ビットが一致するかを調べる。全部一致であればその２文字は一致しているとし、途中で不一致が生じれば、その２文字は不一致とすればよい。

【０１３３】
上述した符号化装置、復号化装置は、それぞれコンピュータにより機能させることもできる。この場合はコンピュータに、目的とする装置(各種実施例で図に示した機能構成を持つ装置)として機能させるためのプログラム、又はその処理手順(各実施例で示したもの)の各過程をコンピュータに実行させるためのプログラムを、ＣＤ－ＲＯＭ、磁気ディスク、半導体記録装置等の記録媒体から、あるいは通信回線を介してそのコンピュータ内にダウンロードし、そのプログラムを実行させればよい。

【０１３４】
［本発明により可変長サンプルを高い圧縮率で符号化できる理由］
以下に、なぜ本発明によれば、音声や音楽などのオーディオ信号、画像信号、各種計測信号、線形予測誤差信号などのディジタル値のサンプル列を、高い圧縮率で符号化できる単位処理情報列に変換できるかを説明する。図３７Ａに、各サンプルのビット数が異なるサンプル列の例（または、有効ビット数が異なるサンプル列の有効ビット部分の例）を示す。このサンプル列を４ビットごとの文字列に変換する場合で検討する。図３７Ｂに、単純に４ビットごとに１文字として文字列に変換した例（従来の方法）を示す。図３７Ｃに、ダミービットを使い、異なるサンプルのビットが１つの文字に含まれないように文字列に変換した例（本発明の方法）を示す。

【０１３５】
まず、本発明の文字列への変換方法によって、高い圧縮率での圧縮符号化が期待できることを示す。
ビット数が可変のサンプル列を文字列（この例では４ビットで１文字）に変換する場合、単純に入力されたビット列を４ビットごとに区切ると、図３７Ｂに示した文字列（ｙ（１），・・・）となる。このように区切ると、例えばｙ（５）は、サンプルｘ（１）の最後（ＬＳＢ側）の２ビットとサンプルｘ（２）の最初（ＭＳＢ側）の２ビットから構成される。

【０１３６】
一方、一般的にオーディオ信号、その他の波形信号などの時系列信号をサンプリングし

て得られたディジタルのサンプル列は、各サンプル間の相関が大きく、各々のサンプルやサンプルの塊が類似していることが多い。ところが、上記のように異なるサンプルのＬＳＢ側とＭＳＢ側のビットから構成される文字を含む文字列に変換してしまうと、各々のサンプルやサンプルの塊が持つ類似性を変換後の文字列にも持たせることができない。したがって、図３７Ｂに示したような単純な変換では、高い圧縮率での符号化が期待できない文字列となってしまう。

【０１３７】

そこで、本発明では、図３７Ｃに示したように、サンプルをまたがった文字列への変換を行わない。本発明では、各サンプルが単位ビット数（この例では４ビット）の整数倍となるまでダミービットを付加する。そして、単位ビットごとに区切ることで文字列に変換する。このように可変長のサンプル列を一定長の文字列に変換すれば、各々のサンプルやサンプルの塊が持つ類似性を変換後の文字列にも持たせることができる。したがって、時系列信号に基づくディジタルサンプル列を、高い圧縮率で圧縮符号化できる。

次に、本発明のマスク処理によって、圧縮率が向上することを示す。

【０１３８】

サンプルごとに有効桁が異なる時系列信号のサンプル列の場合、互いのサンプルの有効桁は異なるが、同一の部分を含んでいるサンプルは多い。しかし、従来の完全に一致する文字を検索する符号化の方法を、有効桁部分のみやダミービットを含んだ文字に対して用いようとすると、完全に一致する文字列が見つからない場合が多い。図３７Ｂの文字列では、連続する複数の文字同士で一致する文字列は存在しない。本発明では、図３７Ｃの文字列のｙ（７），ｙ（８），ｙ（９）を過去の文字列と比較する場合、ｙ（９）の後の３ビットにはマスク処理が行われる。つまり、ｙ（９）の後の３ビット分は比較されない（異なっていても良い）。したがって、ｙ（７），ｙ（８），ｙ（９）とｙ（３），ｙ（４），ｙ（５）とは一致すると判断される。このように、ダミービットとマスク処理とを用いることで、有効桁は異なるが同一の部分が含まれる文字列を同一のものとすることができるため、過去に発生した文字列と同じ文字列が発生する確率が高くなる。したがって、時系列信号に基づくディジタルサンプル列の圧縮効率を高めることができる。

［実験例］

本発明の効果を示すために、図３８Ａと図３８Ｂに、浮動小数点信号の圧縮率を比較した実験の結果を示す。入力信号は、信号１（９６ｋＨｚサンプリング、量子化ビット数２４ビット）、信号２（９６ｋＨｚサンプリング、量子化ビット数２４ビット）、信号３（４８ｋＨｚサンプリング、量子化ビット数２４ビット）の浮動小数点信号である。本発明の方法では、図３７Ｃに示すように、図１５Ｅに相当する整列を行い、８ビットずつを１文字として実施例３の方法で圧縮した。従来法では、実施例３で誤差信号Ｚを符号化する際に、本発明の代わりに図３７Ｂに示すように有効なビットを直列につないで８ビットずつを１文字として従来のＬＺ符号化で符号化した。なお、元の入力信号のサイズをｘ、圧縮後のサイズをｙとすると、従来の圧縮率は、（ｙ／ｘ）×１００としている。図３８Ａ、図３８Ｂから、本発明により高い圧縮率が得られることが分かる。

【図面の簡単な説明】

【０１３９】

【図１】図１Ａは従来のＬＺ７７符号化装置の機能構成を示すブロック図。図１Ｂはその復号化装置の機能構成を示すブロック図。図１Ｃは辞書部及び先読みバッファが連続したスライドバッファで構成されている状態を示す図。

【図２】図２Ａは従来のＬＺ７８符号化装置の機能構成を示すブロック図。図２Ｂはその復号化装置の機能構成を示すブロック図。

【図３】図３Ａは従来のＬＺＷ符号化装置の機能構成を示すブロック図。図３Ｂはその復号化装置の機能構成を示すブロック図。

【図４】ＩＥＥＥ－７５４浮動小数点列を示す図。

【図５】実施例１（実施例Ａ－３）の符号化装置の機能構成例を示すブロック図。

【図６】実施例１（実施例Ａ－３）の符号化装置の処理手順の例を示す流れ図。

【図７】図６中のステップＳ４の具体的処理手順の例を示す流れ図。
【図８】実施例１（実施例Ａ－３）のマスクＬＺ７７及びマスクＬＺ７８の復号化装置の機能構成例を示すブロック図。
【図９】実施例１（実施例Ａ－３）のマスクＬＺ７７及びマスクＬＺ７８の復号化装置の処理手順の例を示す流れ図。
【図１０】実施例１（実施例Ａ－３）のマスクＬＺＷの復号化装置の機能構成例を示すブロック図。
【図１１】実施例１（実施例Ａ－３）のマスクＬＺＷの復号化装置の処理手順の例を示す流れ図。
【図１２】図１２Ａは実施例Ａ－１の処理手順の一部を示す流れ図。図１２Ｂは実施例Ａ－１の評価部４２ｃの機能構成例を示すブロック図。
【図１３】図１３Ａは実施例Ａ－４のマスクＬＺＷにおける辞書部４３及びマスク計算部４８′の例を示すブロック図。図１３Ｂは実施例Ａ－４のマスクＬＺＷにおける符号生成部４９の一部の具体的機能構成例を示すブロック図。
【図１４】図１４Ａは、実施例Ａ－１乃至Ａ－３におけるマスクＬＺ７７の辞書部４３と先読みバッファ６５及び有効桁情報バッファとの関係例を示す図。図１４Ｂは、実施例Ａ－４及びＡ－５におけるマスクＬＺ７７の辞書部と先読みバッファと有効桁情報バッファとの関係例を示す図。
【図１５】図１５Ａは、可変長サンプルごとにビット数が異なる場合の入力サンプル列の例を示す図。図１５Ｂは、各入力サンプルをＬＳＢ側に整列し、単位処理情報へ変換した例を示す図。図１５Ｃは、各入力サンプルをＭＳＢ側に整列し、単位処理情報へ変換した例を示す図。図１５Ｄは、各入力サンプルをＬＳＢ側に整列し、単位処理情報のビット数N_Cずつに切り出した例を示す図。図１５Ｅは、各入力サンプルをＭＳＢ側に整列し、単位処理情報のビット数N_Cずつに切り出した例を示す図。図１５Ｆは、各入力サンプルをＬＳＢ側に整列し、有効桁よりもＭＳＢ側にはダミービットを付加してビット数を拡張する例を示す図。図１５Ｇは、各入力サンプルをＭＳＢ側に整列し、有効桁よりもＬＳＢ側にはダミービットを付加してビット数を拡張する例を示す図。
【図１６】図１６Ａは、各入力サンプルが一定の大きなビット長であり、且つ、各サンプルの有効桁数がビット長に比べて短い場合の、マスクＬＺ方法の単位処理情報の変換例を示す図。図１６Ｂは、サンプル内の有効なビットをＬＳＢ側に揃えて、ダミービットのマスクを設定する例を示す図。図１６Ｃは、サンプル内の有効なビットをＭＳＢ側に揃えて、誤差を許容する部分にダミービットのマスクを設定する例を示す図。図１６Ｄは、各入力サンプルの有効なビットをＬＳＢ側に揃え、サンプル語長よりも短いビット数の単位処理情報のビット数ごとに、サンプルから有効なビットの部分を切り出す方法を示す図。図１６Ｅは、各入力サンプルの有効なビットをＭＳＢ側に揃え、サンプル語長よりも短いビット数の単位処理情報のビット数ごとに、サンプルから有効なビットの部分を切り出す方法を示す図。
【図１７】実施例２の符号化装置の機能構成例を示すブロック図。
【図１８】実施例２の単位処理列変換部７９の処理手順の例を示す流れ図。
【図１９】実施例２の復号化装置の機能構成例を示すブロック図。
【図２０】実施例２のサンプル列変換部８４の処理手順の例を示す流れ図。
【図２１】実施例３の符号化装置の機能構成例を示すブロック図。
【図２２】実施例３の符号化装置の処理手順の例を示す流れ図。
【図２３】各種サンプルの整数形式信号とその差分信号の関係例を示す図。
【図２４】実施例３の復号化装置の機能構成例を示すブロック図。
【図２５】実施例３の復号化装置の処理手順の例を示す流れ図。
【図２６】複数の二進表現数に対し共通の乗数で割算することにより符号化効率をあげるように変換する具体的数値例を示す図。
【図２７】複数の浮動小数点数に対し共通の乗数で割算することにより符号化効率をあげるように変換する具体的数値例を示す図。

【図２８】実施例４の符号化装置の機能構成例を示すブロック図。
【図２９】実施例４の符号化装置の処理手順の例を示す流れ図。
【図３０】実施例４の復号化装置の機能構成例を示すブロック図。
【図３１】実施例４の復号化装置の処理手順の例を示す流れ図。
【図３２】実施例５の符号化装置の機能構成例を示すブロック図。
【図３３】実施例５の符号化装置の処理手順の例を示す流れ図。
【図３４】実施例５の復号化装置の機能構成例を示すブロック図。
【図３５】図３５Ａは、単位処理情報列の符号化部を変形した部分の機能構成例を示すブロック図。図３５Ｂは、単位処理列変換の一部を変形した手順例を示す流れ図。
【図３６】最長一致文字探索にハッシュ関数処理を適用した機能構成の一部を示すブロック図。
【図３７】図３７Ａは、各サンプルのビット数が異なるサンプル列の例を示す図。図３７Ｂは、単純に４ビットごとに１文字として文字列に変換した例を示す図。図３７Ｃは、ダミービットを使い、サンプル間の相関を維持して文字列に変換した例を示す図。
【図３８】図３８Ａは、浮動小数点信号を従来法と本発明の方法で圧縮した場合の圧縮結果を示す図。図３８Ｂは、浮動小数点信号の圧縮率を比較する図。

【図１】

図1A

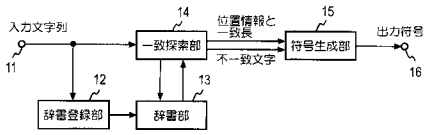


図1B

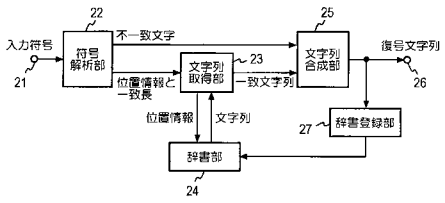
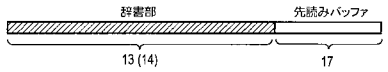


図1C



【図２】

図2A

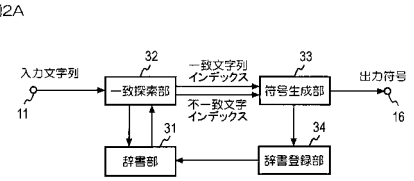
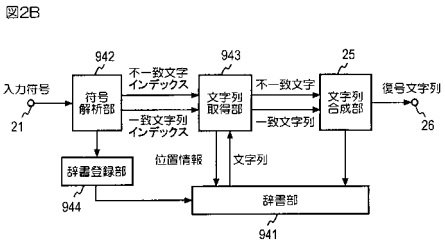


図2B



【図３】

図3A

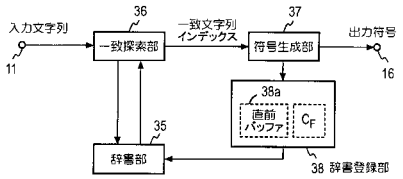
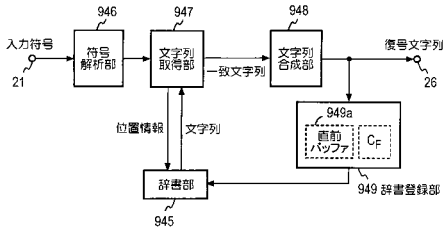
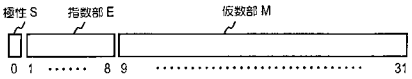


図3B



【図４】

図4



【図５】

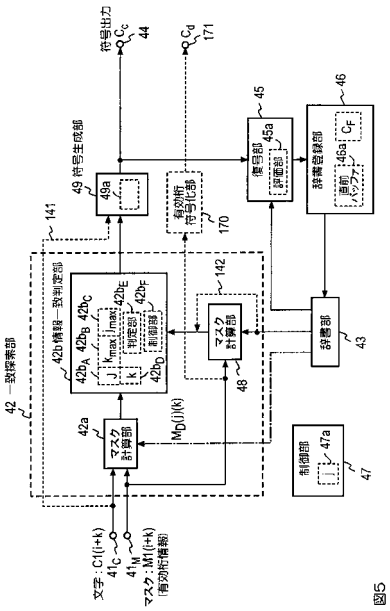
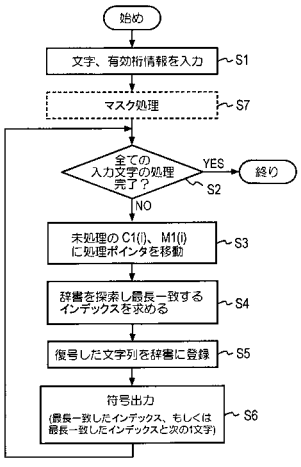


図5

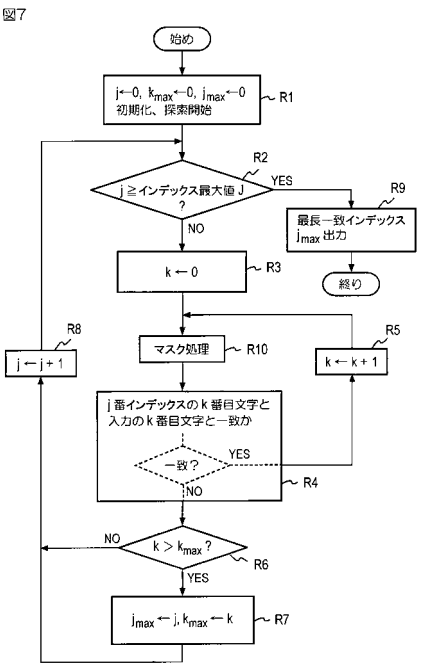
【図６】

図6

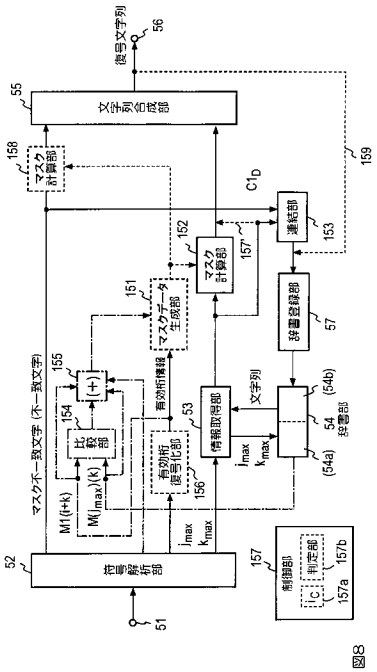


【図７】

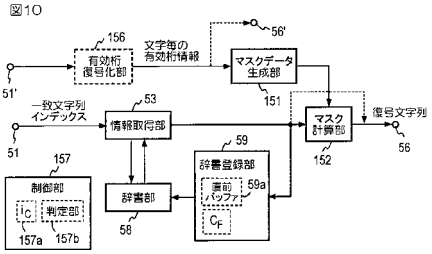
図7



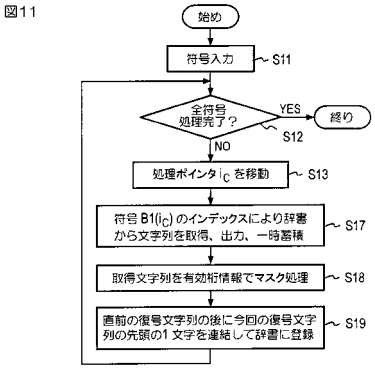
【図８】



【図１０】



【図１１】



【図１２】

図12A

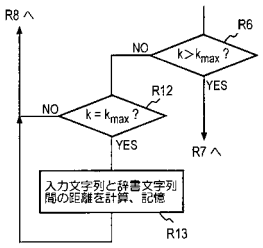
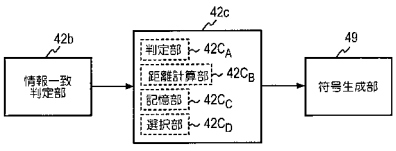


図12B



【図１３】

図13A

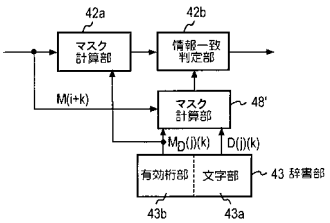
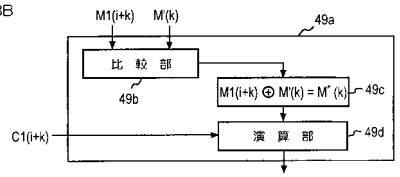


図13B



【図１４】

図14A

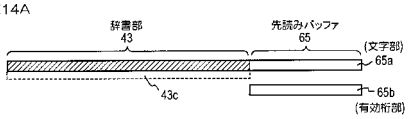
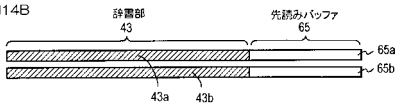
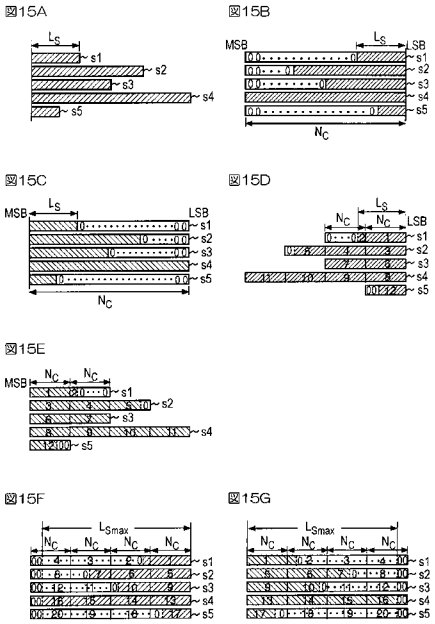


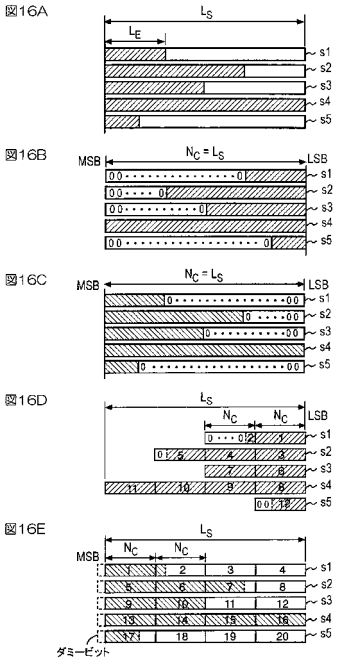
図14B



【図１５】



【図１６】



【図１７】

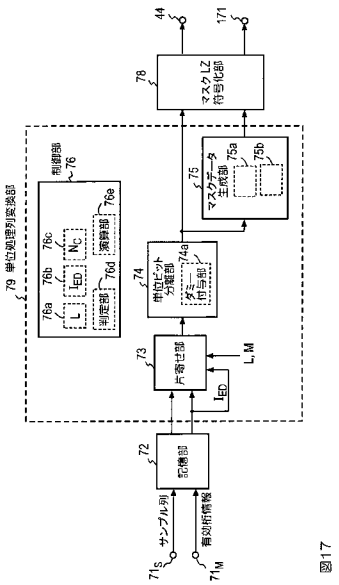
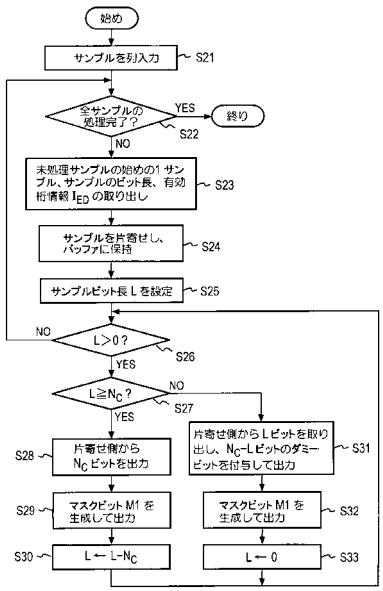


図17

【図１８】

図18



【図１９】

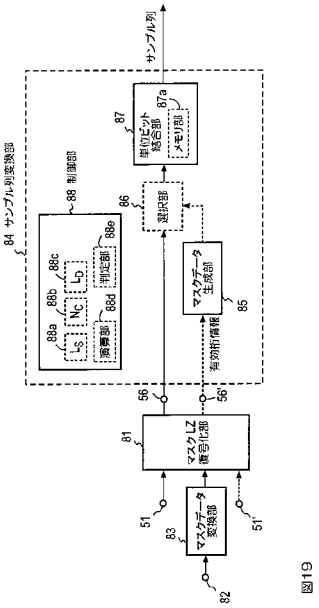
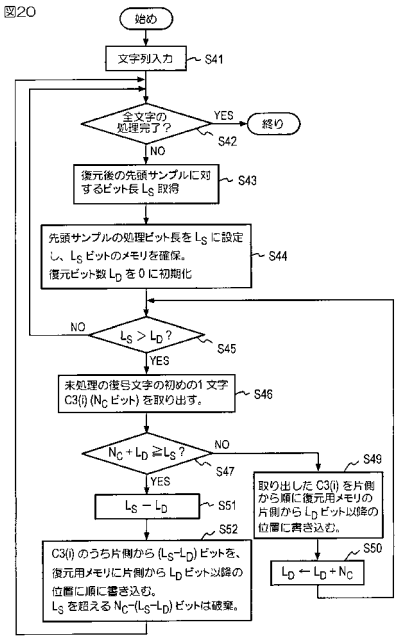


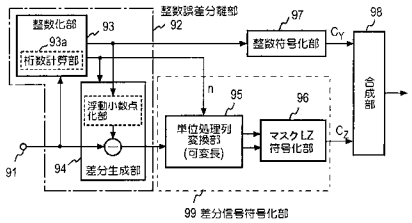
図19

【図２０】



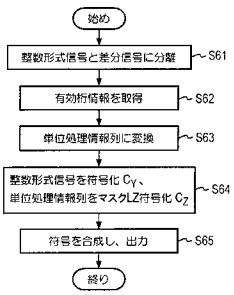
【図２１】

図21



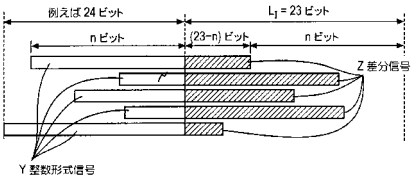
【図２２】

図22



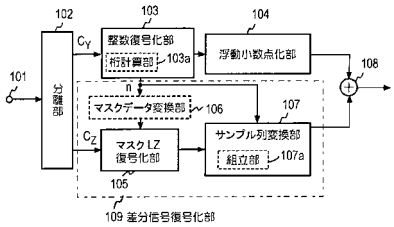
【図２３】

図23



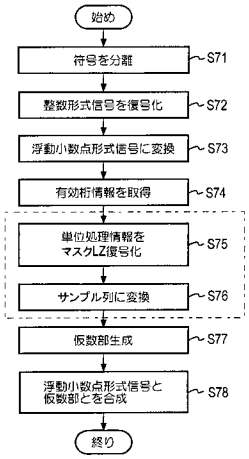
【図２４】

図24



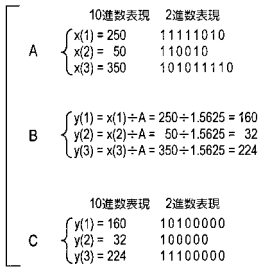
【図２５】

図25



【図２６】

図26



【図２７】

図27

【図２８】

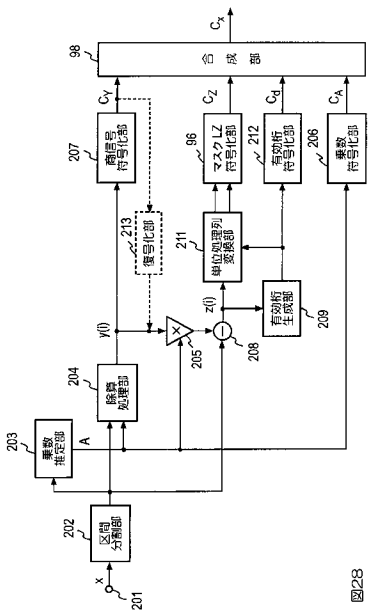
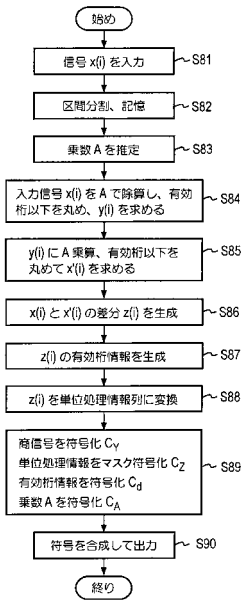


図28

【図２９】

図29



【図３０】

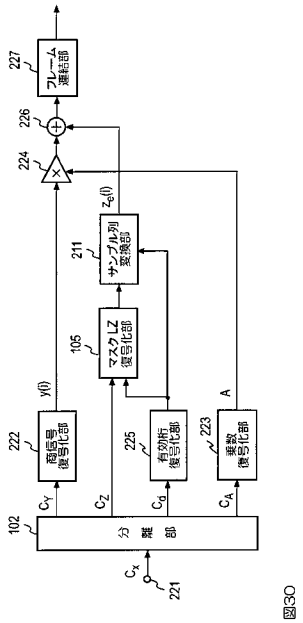
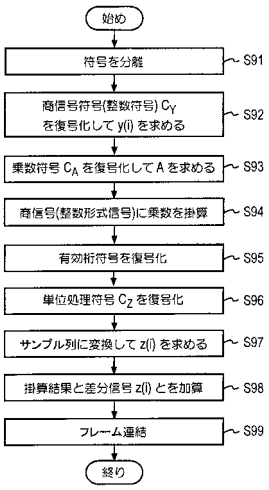


図30

【図３１】

図31



【図３２】

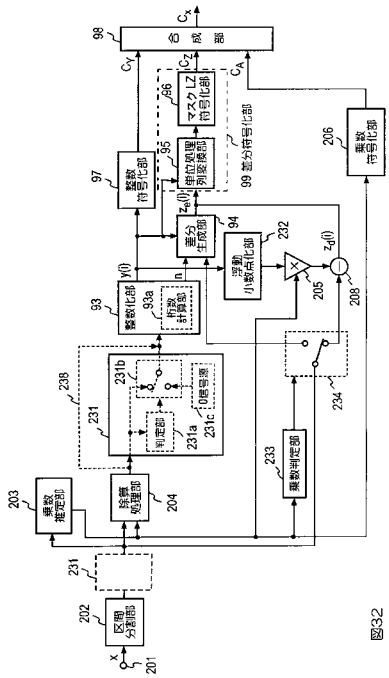
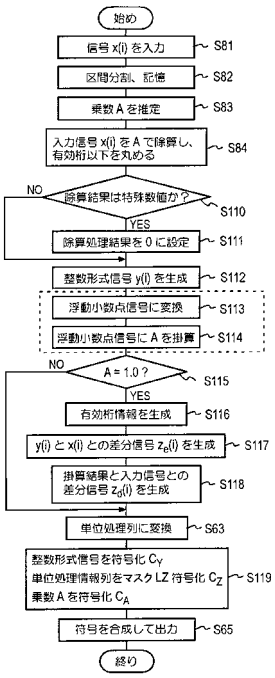


図32

【図３３】

図33



【図３４】

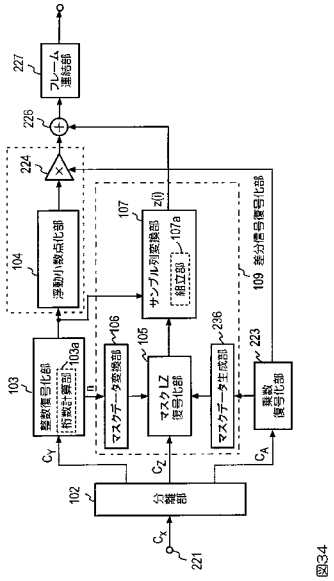


図34

【図３５】

図35A

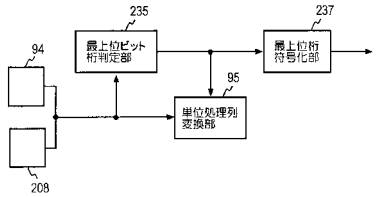
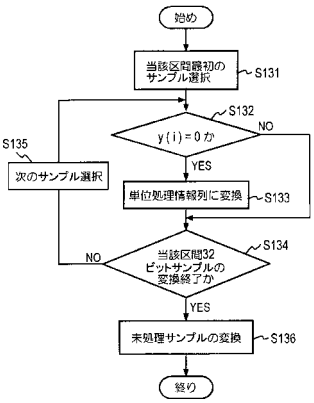
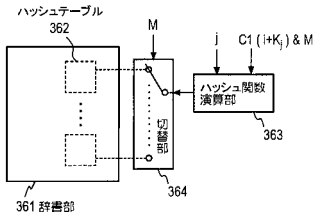


図35B



【図３６】

図36



【図３７】

図37A サンプル例 {x(1),…}

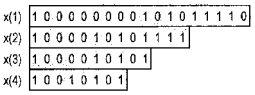


図37B 文字列 {y(1),…}

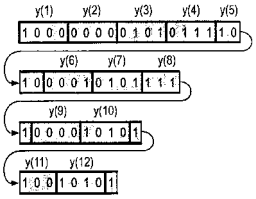
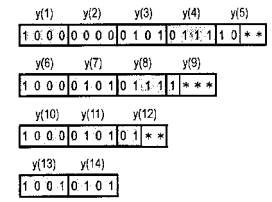


図37C 文字列 {y(1),…}

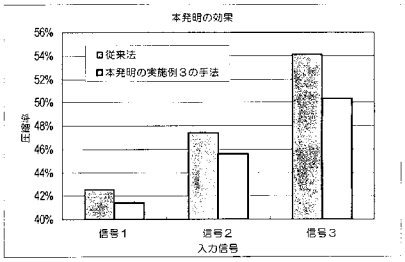


【図３８】

図38A

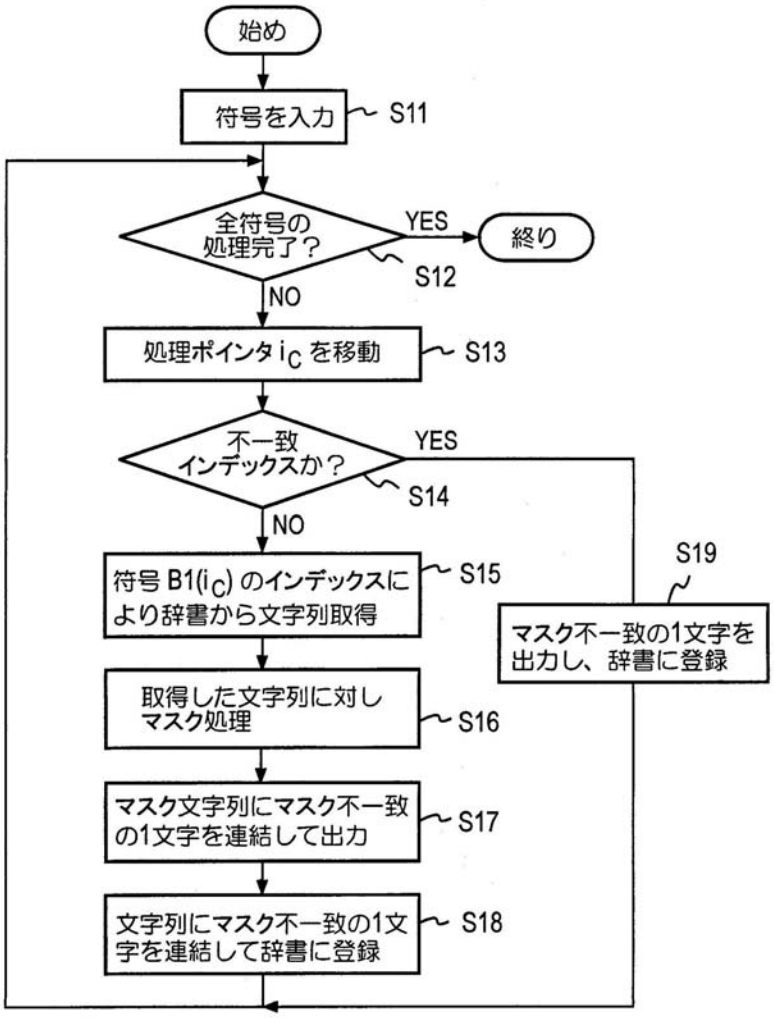
入力信号		元のサイズ	従来法		本発明	
			圧縮後のサイズ	圧縮率	圧縮後のサイズ	圧縮率
信号1	96kHz, 24bit	66,007,332	28,091,208	42.558%	27,346,043	41.429%
信号2	96kHz, 24bit	24,465,528	11,593,255	47.386%	11,152,211	45.583%
信号3	48kHz, 24bit	11,520,244	6,230,924	54.087%	5,798,421	50.332%

図38B



【図９】

図9


始め
符号を入力
S11
全符号の処理完了？
YES
終り
S12
NO
処理ポインタ i_C を移動
S13
不一致インデックスか？
YES
S14
NO
符号 B1(i_C) のインデックスにより辞書から文字列取得
S15
取得した文字列に対しマスク処理
S16
マスク文字列にマスク不一致の1文字を連結して出力
S17
文字列にマスク不一致の1文字を連結して辞書に登録
S18
S19
マスク不一致の1文字を出力し、辞書に登録

フロントページの続き

審査官　北村　智彦

(56)参考文献　特開２００３－１７９５００（ＪＰ，Ａ）
特開平１０－１９０４７６（ＪＰ，Ａ）
特開平５－１２７８６６（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
H03M3/00-11/00