(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6045123号
(P6045123)
(45)発行日　平成28年12月14日(2016.12.14)　(24)登録日　平成28年11月25日(2016.11.25)

(51)Int.Cl.　　F I
H03M 7/36 (2006.01)　　H03M 7/36
H03M 7/40 (2006.01)　　H03M 7/40

請求項の数 29　(全 39 頁)

(21)出願番号　特願2015-559442 (P2015-559442)
(86)(22)出願日　平成26年2月27日 (2014.2.27)
(65)公表番号　特表2016-513436 (P2016-513436A)
(43)公表日　平成28年5月12日 (2016.5.12)
(86)国際出願番号　PCT/EP2014/000510
(87)国際公開番号　WO2014/131517
(87)国際公開日　平成26年9月4日 (2014.9.4)
審査請求日　平成27年8月28日 (2015.8.28)
(31)優先権主張番号　1303661.1
(32)優先日　平成25年3月1日 (2013.3.1)
(33)優先権主張国　英国 (GB)

早期審査対象出願

(73)特許権者　513156386
グルロジック　マイクロシステムズ　オーワイ
Ｇｕｒｕｌｏｇｉｃ　Ｍｉｃｒｏｓｙｓｔｅｍｓ　Ｏｙ
フィンランド共和国　２０１００　トゥルク　リンナンカツ　３４
Ｌｉｎｎａｎｋａｔｕ　３４　２０１００　Ｔｕｒｋｕ　ＦＩＮＬＡＮＤ
(74)代理人　100127188
弁理士　川守田　光紀
(72)発明者　カレヴォ　オッシ
フィンランド共和国　ＦＩＮ－３７８００　トイヤラ　ケトゥンハンタ　１

最終頁に続く

(54)【発明の名称】エンコーダ、デコーダ及び方法

(57)【特許請求の範囲】
【請求項１】
数値の列を含む入力データ（Ｄ１）を符号化することにより、対応符号化出力データ（Ｄ２又はＤ３）を生成する方法であって、
（Ａ）前記入力データ中の特定の数値に対して、予測値を減算することにより出力値を計算すること、ただし前記減算の結果が、前記入力データ中の数値の最小値未満となった場合には、該結果に更にラップアラウンド値を加算することにより、前記出力値を計算すること；または、
（Ｂ）前記入力データ中の特定の数値に対して、予測値を加算することにより出力値を計算すること、ただし前記加算の結果が、前記入力データ中の数値の最大値を超える場合には、該結果からラップアラウンド値を減算することにより、前記出力値を計算すること；
を含む、方法。
【請求項２】
前記ラップアラウンド値は、highValue - lowValue + 1であり、ここでhighValueは前記入力データ中の数値の最大値であり、lowValueは前記入力データ中の数値の最小値である、請求項１に記載の方法。
【請求項３】
前記予測値は、前記入力データの中で最初に処理する数値に対してはデフォルト予測値であり、前記入力データのその他の数値に対しては、直前に処理した数値に等しい、請求項１に記載の方法。

【請求項４】
前記予測値は、前記入力データの中で最初に処理する数値に対してはデフォルト予測値であり、前記入力データのその他の数値に対しては、直前に処理した数値から得られた前記出力値に等しい、請求項１に記載の方法。
【請求項５】
前記デフォルト予測値は、
０；
highValue/2；
highValue/2 + lowValue；
highValue - lowValue；
（highValue + lowValue + 1）/2；
の少なくともいずれかであり、ただしhighValueは前記入力データ中の数値の最大値であり、lowValueは前記入力データ中の数値の最小値である、請求項３または４に記載の方法。
【請求項６】
前記入力データ中の数値の最小値を０にするようなオフセット値を前記入力データ中の各数値に適用することを更に含む、請求項２に記載の方法
【請求項７】
前記オフセット値をデコーダに伝達することを含む、請求項６に記載の方法。
【請求項８】
前記入力データ中の数値の最大値であるhighValueと、前記入力データ中の数値の最小値であるlowValueとを示す情報をデコーダに伝達することを更に含む、請求項１から７のいずれかに記載の方法。
【請求項９】
前記対応符号化出力データを前記入力データとして、前記（Ａ）または前記（Ｂ）を繰り返し遂行することを更に含み、ただし、前記繰り返しの間に前記（Ａ）及び前記（Ｂ）をそれぞれ少なくとも１回以上遂行する、請求項１から８のいずれかに記載の方法。
【請求項１０】
前記数値は複数のビットで表される、請求項１から９のいずれかに記載の方法。
【請求項１１】
１つ又は複数の１ビット値を含む数値を処理することを含み、
ビット単位方式で前記入力データ（Ｄ１）を符号化することを含む、
請求項１から９のいずれかに記載の方法。
【請求項１２】
別々に符号化される複数のデータセクションに前記入力データ（Ｄ１）を分割することを更に含む、請求項１から１１のいずれかに記載の方法。
【請求項１３】
前記データセクションに対して選択的に符号化を適用することを含み、該符号化を適用することは、これによってデータ圧縮が符号化出力データ（Ｄ２又はＤ３）において達成可能である場合にのみ実行されるものである、請求項１２に記載の方法。
【請求項１４】
更なる符号化を適用することにより、符号化出力データ（Ｄ２又はＤ３）を生成することを含み、該更なる符号化が、ランレングス符号化（ＲＬＥ）、可変長符号化（ＶＬＣ）、ハフマン符号化、算術符号化、レンジ符号化のうちの少なくとも１つを含む、請求項１から１３の何れか１項に記載の方法。
【請求項１５】
相互に類似のビットのランレングスに応じて前記入力データ（Ｄ１）を複数のセクションに分割することを更に含み、前記ランレングスは、ランレングス符号化（ＲＬＥ）、ハフマン符号化、可変長符号化（ＶＬＥ）、レンジ符号化及び／又は算術符号化を用いる符号化に有効なものである、請求項１２に記載の方法。

【請求項１６】
処理手段及び記憶手段を備える装置であって、前記記憶手段はプログラム命令を格納し、該プログラム命令は、前記処理手段に実行されると、請求項１から１５のいずれかに記載の方法を遂行させるように構成される、装置。
【請求項１７】
装置の処理手段に実行されると、前記装置に、請求項１から１５のいずれかに記載の方法を遂行させるように構成されるプログラム命令を備える、コンピュータプログラム。
【請求項１８】
符号化データ（Ｄ２又はＤ３）を復号するデコーダ（２０）を用いることにより対応復号出力データ（Ｄ５）を生成する方法であって、
（Ａ）前記符号化データ中の特定の数値に対して、予測値を加算することにより出力値を計算すること、ただし前記加算の結果が、前記符号化データ中の数値の最大値を超える場合には、該結果からラップアラウンド値を減算することにより、前記出力値を計算すること；または、
（Ｂ）前記符号化データ中の特定の数値に対して、予測値を減算することにより出力値を計算すること、ただし前記減算の結果が、前記符号化データ中の数値の最小値未満となった場合には、該結果に更にラップアラウンド値を加算することにより、前記出力値を計算すること；
を含む、方法。
【請求項１９】
前記ラップアラウンド値は、highValue － lowValue ＋ 1であり、ここでhighValueは前記符号化データ中の数値の最大値であり、lowValueは前記符号化データ中の数値の最小値である、請求項１８に記載の方法。
【請求項２０】
前記予測値は、前記符号化データの中で最初に処理する数値に対してはデフォルト予測値であり、前記符号化データのその他の数値に対しては、直前に処理した数値から得られた前記出力値に等しい、請求項１８に記載の方法。
【請求項２１】
前記予測値は、前記符号化データの中で最初に処理する数値に対してはデフォルト予測値であり、前記符号化データのその他の数値に対しては、直前に処理した数値に等しい、請求項１８に記載の方法。
【請求項２２】
前記デフォルト予測値は、
０；
highValue/2；
highValue/2 ＋ lowValue；
highValue － lowValue；
（highValue ＋ lowValue ＋ 1）/2；
の少なくともいずれかであり、ただしhighValueは前記符号化データ中の数値の最大値であり、lowValueは前記符号化データ中の数値の最小値である、請求項２０または２１に記載の方法。
【請求項２３】
前記対応復号出力データを前記符号化データとして、前記（Ａ）または前記（Ｂ）を繰り返し遂行することを更に含み、ただし、前記繰り返しの間に前記（Ａ）及び前記（Ｂ）をそれぞれ少なくとも１回以上遂行する、請求項１８から２２のいずれかに記載の方法。
【請求項２４】
前記数値は複数のビットで表される、請求項１８から２３のいずれかに記載の方法。
【請求項２５】
１つ又は複数の１ビット値を含む前記符号化データ（Ｄ２又はＤ３）を復号することを含み、

ここで前記デコーダ（１９）は、ビット単位方式で前記符号化データ（Ｄ２又はＤ３）を復号するように動作可能である、
請求項１８から２３のいずれかに記載の方法。
【請求項２６】
前記ステップ（Ａ）又はステップ（Ｂ）を行う前に、前記符号化データに対して、ランレングス符号化（ＲＬＥ）、可変長符号化（ＶＬＣ）、ハフマン符号化、算術符号化、レンジ符号化の少なくとも１つの逆を適用することを含む、請求項１８から２５のいずれかに記載の方法。
【請求項２７】
処理手段及び記憶手段を備える装置であって、前記記憶手段はプログラム命令を格納し、該プログラム命令は、前記処理手段に実行されると、請求項１８から２６のいずれかに記載の方法を遂行させるように構成される、装置。
【請求項２８】
装置の処理手段に実行されると、前記装置に、請求項１８から２６のいずれかに記載の方法を遂行させるように構成されるプログラム命令を備える、コンピュータプログラム。
【請求項２９】
請求項１６に記載の装置を備えるエンコーダを少なくとも一つと、請求項２７に記載の装置を備えるデコーダを少なくとも一つとを有する、コーデック装置（３０）。
【発明の詳細な説明】
【技術分野】
【０００１】
本開示は、エンコーダに関し、例えば直接ＯＤｅｌｔａ演算（direct ODelta operator）を利用するよう動作可能であるエンコーダ等に関する。さらに、本開示は、データを符号化する方法に関し、例えば直接ＯＤｅｌｔａ演算を利用してデータを符号化する方法に関する。さらに、本開示は、符号化されたデータを復号するデコーダにも関し、例えば逆ＯＤｅｌｔａ演算を適用するよう動作可能であるデコーダに関する。加えて、本開示は、符号化されたデータを復号する方法に関連し、例えば逆ＯＤｅｌｔａ演算を適用することによって符号化されたデータを復号する方法に関する。さらに加えて、本開示は、不揮発性（非一時的）機械可読データ記憶媒体に記録されたソフトウェア製品であって、上記方法を実装するコンピュータハードウェアで実行可能であるソフトウェア製品に関する。
【背景】
【０００２】
クロード・Ｅ・シャノン（Ｃｌａｕｄｅ Ｅ．Ｓｈａｎｎｏｎ）は、現代の通信システムの基礎を提供する、通信の数学的理論を提唱した。さらに、様々な現代の符号化法が上記数学的理論の知識において発展してきた。表１には、現代の技術知識の概要を提供する情報源のリストが示されている。
【０００３】

表１： 既知の技術

先行文献	詳細
P1	可変長符号（“Variable-length code”）、ウィキペディア（2012年11月28日にアクセス） URL: http://en.wikipedia.org/wiki/Variable-length_code
P2	ランレングス符号化（“Run-length encoding”）、ウィキペディア（2012年11月28日にアクセス） URL: http://en.wikipedia.org/wiki/Run-length_encoding
P3	ハフマン符号化（“Huffman coding”）、ウィキペディア（2012年11月28日にアクセス） URL: http://en.wikipedia.org/wiki/Huffman_coding
P4	算術符号化（“Arithmetic coding”）、ウィキペディア（2012年11月28日にアクセス） URL: http://en.wikipedia.org/wiki/Arithmetic_coding
P5	通信の数学的理論（“A Mathematic Theory of Communication”）、シャノン・クロード・E（Shannon, Claude E.）（１９４８）、ウィキペディア（2012年11月28日にアクセス） URL: http://cm:bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf
P6	差分符号化（“Delta encoding”）、ウィキペディア（2012年11月28日にアクセス） URL: http://en.wikipedia.org/wiki/Delta_coding
P7	シャノンの情報源符号化定理（Shannon’s source coding theorem）；ウィキペディア（2012年11月28日にアクセス） URL: http://en.wikipedia.org/wiki/Source_coding_theorem
P8	エントロピー（“Entropy”） ウィキペディア（2012年11月28日にアクセス） URL: http://en.wikipedia/wiki/Entropy

【０００４】

欧州特許ＥＯ１３７６９７４Ｂ１（「パケットヘッダ圧縮の方法及び装置」、アルカテル・ルーセント（Alcatel Lucent）（フランス））には、データパケットを送信する方法であって、データパケットが、圧縮値を含むフィールド（ＣＦ）を含む、方法が記載されている。この圧縮値は、２つの連続データパケットの間で生じる値を示すものであり、所定の区間（以下、「解析区間」（”interpretation interval”））を含む。この方法は、以下の工程を含む。

【０００５】

（ｉ）圧縮される値及と所定のラップアラウンド境界（wraparound boundary）との間の距離が、所定の閾値よりも小さい場合に、圧縮された値に対する追加ビットであって、ラップアラウンド境界に対する、圧縮される値の相対位置を一義的に示す追加ビットを付加すること、

（ｉｉ）レシーバにおいて、圧縮値の受信ビット全てに応じて第１解析区間を計算すること、
（ｉｉｉ）第１解析区間の１を超える値が受信圧縮値に一致した場合、ラップアラウンドを信号伝達すること、
（ｉｖ）ラップアラウンドの信号伝達を受けて、上記追加ビットを除く圧縮値の受信ビット全てに従い、第２解析区間を計算すること、並びに
（ｖ）上記追加ビットを用いて、上記第２解析区間における展開値について曖昧性解消を実行すること。

【０００６】

シャノンエントロピーの定義は、表１に記載される文献Ｐ７及びＰ８で与えられる。所与のデータに存在するエントロピーを圧縮するよう動作可能である各種圧縮法が多数存在し、それら方法は、例えば所与のデータに対してより大きい可逆圧縮率を獲得することを目的としてエントロピーを変更するために時として利用される。このようなエントロピー変更方法は、例えば、表１の文献Ｐ２に記載されるランレングス符号化（run-length-encoding；ＲＬＥ）、表１の文献Ｐ１に記載される可変長符号化（variable-length-coding；ＶＬＣ）、表１の文献Ｐ３に記載のハフマン符号化（Huffman coding）、表１の文献Ｐ６に記載の差分符号化（Delta coding）、及び表１の文献Ｐ４に記載の算術符号化（Arithmetic coding）、即ちレンジ符号化（Range coding）を含むものである。これら方法は、アルファベット文字、数字、バイト及びワードを表すデータを圧縮するために有利に利用される。しかしながら、これら方法は、ビットレベルにおいて所与のデータを圧縮するのに十分に適合されたものではなく、そのため、ビット単位で変化しやすい上記のような所与のデータを圧縮するのに十分に適したものとはいえない。

【０００７】

例えば表１の文献Ｐ６に記載されるような差分符号化は、正の元データ値から正又は負の値を取り得る差分値を生成するように機能可能である。さらに、用いられるデータ要素の大きさいに基づいて８ビット、１６ビット又は３２ビットのラップアラウンドに用いるための差分エンコーダの実装が知られている。しかしながら、８ビット、１６ビット又は３２ビットの値以外のレジーム用に最適化された差分エンコーダは、現在、存在しない。詳細には、元ビット値、即ち「０」及び「１」を、例えばビット単位の符号化において、それら値から典型的に生じる３つの異なる値、即ち「－１」、「０」及び「１」として符号化する場合には、既知の差分エンコーダは特に効率が悪い。

【０００８】

あらゆる種類のデータが記憶領域を消費し、また上記のようなデータをある位置から他の位置に移動する際に通信システム伝送容量が必要とされる。例えば３次元動画コンテンツ等のマルチメディアの発展の結果、データ量が増加するにつれ、それに応じて、より多くの記憶領域及び伝送容量が、当該データを取り扱うのに必要とされ、さらに、データ量が増加するにつれ、より多くのエネルギーが必要とされる。世界規模で、伝送されるデータ量は、時代と共に着実に増加しており、例えば、インターネットは、膨大な量のデータを包含しており、そのデータの一部は、複数コピーで記憶されている。さらに、例えばデータのサイズを小さくする際に用いられる、該データに関連するエントロピーＥを圧縮するために現在利用可能な方法がある。さらに、例えば差分符号化及びランレングス符号化（run-length encoding；ＲＬＥ）等、エントロピーを変更する方法もまた利用可能なものがあるが、現在、実現可能なものよりも更に高い程度のデータ圧縮を提供するために、改良法が必要とされている。

【０００９】

例えば元データのデータ要素における全ての値が用いられず、かつ／又は、差分符号化法と組み合わせて利用される先行又は後続の符号化法が、符号化されるデータに当初用いられるビットダイナミックよりも高いビットフォーマットを必要とする場合は、より高速かつより効率的な元データの符号化を達成することを可能とするために既知の差分符号化法、例えばビット単位の符号化の利用を最適化する必要もある。

【先行技術文献】
【特許文献】
【００１０】
【特許文献１】欧州特許ＥＯ１３７６９７４Ｂ１
【非特許文献】
【００１１】
【非特許文献１】可変長符号（”Variable-length code”）、ウィキペディア（2012年11月28日にアクセス）URL: http://en.wikipedia.org/wiki/Variable-length_code
【非特許文献２】ランレングス符号化（”Run-length encoding”）、ウィキペディア（2012年11月28日にアクセス）URL: http://en.wikipedia.org/wiki/Run-length_encoding
【非特許文献３】ハフマン符号化（”Huffman coding”）、ウィキペディア（2012年11月28日にアクセス）URL: http://en.wikipedia.org/wiki/Huffman_coding
【非特許文献４】算術符号化（”Arithmetic coding”）、ウィキペディア（2012年11月28日にアクセス）URL: http://en.wikipedia.org/wiki/Arithmetic_coding
【非特許文献５】通信の数学的理論（”A Mathematic Theory of Communication”）、シャノン・クロード・Ｅ（Shannon, Claude E.）（１９４８）、ウィキペディア（2012年11月28日にアクセス）URL: http://cm:bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf
【非特許文献６】差分符号化（”Delta encoding”）、ウィキペディア（2012年11月28日にアクセス）URL: http://en.wikipedia.org/wiki/Delta_coding
【非特許文献７】シャノンの情報源符号化定理（Shannon’s source coding theorem）；ウィキペディア（2012年11月28日にアクセス）URL: http://en.wikipedia.org/wiki/Source_coding_theorem
【非特許文献８】エントロピー（”Entropy”）- ウィキペディア（2012年11月28日にアクセス）URL: http://en.wikipedia/wiki/Entropy
【摘要】
【００１２】
本開示は、各ビットそれぞれを符号化する場合、つまりビット単位の符号化において各ビットを符号化する場合、並びにデータのその他値を符号化する場合により効率的な、差分エンコーダの改良型、即ち「直接ＯＤｅｌｔａエンコーダ」を提供することを目的とする。
【００１３】
また、本開示は、データを差分符号化する改良法、即ち、データを直接ＯＤｅｌｔａ符号化する方法を提供することを目的とする。
【００１４】
さらに、本開示は、符号化データ（例えば、ＯＤｅｌｔａ符号化データ）を復号する改良デコーダ、即ち、逆ＯＤｅｌｔａデコーダを提供することを目的とする。
【００１５】
加えて、本開示は、符号化データ（例えば、ＯＤｅｌｔａ符号化データ）を復号する改良法、即ち、データの逆ＯＤｅｌｔａ復号法を提供することを目的とする。
【００１６】
第１の態様によれば、添付の請求項１に記載されるエンコーダが提供される。即ち、数値列を含む入力データ（Ｄ１）を符号化することにより、対応符号化出力データ（Ｄ２又はＤ３）を生成するエンコーダ（１０）であって、エンコーダ（１０）は、差分符号化及び／又は合算符号化の方式を入力データ（Ｄ１）に適用することにより１つ又は複数の対応符号化列を生成するデータ処理装置を備えるものであり、但し、上記１つ又は複数の対応符号化列は、符号化出力データ（Ｄ２又はＤ３）を生成するために、最大値におけるラップアラウンド及び／又は最小値におけるラップアラウンドを受け、かつエンコーダ（１０）は、出力符号化データ（Ｄ２又はＤ３）を作成するのに利用される一連の予測値としてデフォルト第１予測値を利用するように動作可能であり、該符号化出力データ（Ｄ２又

はＤ３）は、入力値、予測値及び符号化演算を用いて生成されることを特徴とする、エンコーダ（１０）が提供される。

【００１７】

上記ラップアラウンドは、上記エンコーダを実施する場合に、負の差分又は極端に大きい値を生じさせないために利用されるという利点を持つ。「数値」という用語は、個々のビット（２進法）、並びにビットの群（２進数より高次）を含むものと解される。

【００１８】

本発明は、差分符号化及び／又は合算符号化並びにラップアラウンドの組合せにより、符号化データ（Ｄ２）を生成する上で入力データ（Ｄ１）の有用なエントロピー変更を提供することが可能であり、この事は、符号化データ（Ｄ２）から符号化データ（Ｄ３）を生成する際に所与のエントロピーエンコーダ内でデータ圧縮強化の達成を可能にするという利点を有する。

【００１９】

オプションで、上記エンコーダに関し、上記データ処理装置は、入力データ（Ｄ１）及び／又は上記１つ又は複数の対応符号化列を分析することにより、符号化出力データ（Ｄ２又はＤ３）の生成に用いるための上記１つ又は複数の対応符号化列に適用する１つ又は複数のオフセット値、最小値、及び／又は最大値を計算するように動作可能である。さらにオプションで、上記エンコーダに関し、上記１つ又は複数のオフセット値は、値「０」を有する。

【００２０】

オプションで、上記エンコーダは、１つ又は複数の１ビット値を含む数値を処理するように動作可能であり、かつ上記エンコーダは、ビット単位方式で入力データ（Ｄ１）を符号化するように動作可能である。

【００２１】

オプションで、上記エンコーダにおいて、上記１つ又は複数の対応符号化列は、入力データ（Ｄ１）の連続値における変化を表す。

【００２２】

オプションで、上記エンコーダは、ラップアラウンド値（ｗｒａｐＶａｌｕｅ）を利用するように動作可能であり、該ラップアラウンド値は、最大値（ｈｉｇｈＶａｌｕｅ）－最小値（ｌｏｗＶａｌｕｅ）＋１である。このようなラップアラウンド値は、符号化出力データ（Ｄ２又はＤ３）を生成する際に、最小値（ｌｏｗＶａｌｕｅ）未満の値や、しばしば負の値、又は最大値（ｈｉｇｈＶａｌｕｅ）を超える値を生じさせないために、有利に用いられる。

【００２３】

オプションで、上記エンコーダは、別々に符号化される複数のデータセクションに入力データ（Ｄ１）を分割するように動作可能である。さらにオプションで、上記エンコーダは、上記データセクションに対して選択的に符号化を適用するように動作可能であり、該符号化の適用は、これによってデータ圧縮が符号化出力データ（Ｄ２又はＤ３）において達成可能である場合にのみ実行されるものである。

【００２４】

さらにオプションで、上記エンコーダにおいて、デフォルト第１予測値は「０」である。さらにオプションで、上記第１予測値は、また、最大値（ｈｉｇｈＶａｌｕｅ）、最小値（ｌｏｗＶａｌｕｅ）、（最大値（ｈｉｇｈＶａｌｕｅ）＋最小値（ｌｏｗＶａｌｕｅ）÷２等であり得る。

【００２５】

オプションで、上記エンコーダは、更なる符号化を適用することにより、符号化入力データ（Ｄ２又はＤ３）を生成し、該更なる符号化が、ランレングス符号化（ＲＬＥ）、分割ＲＬＥ、エントロピー変更（ＥＭ）、可変長符号化（ＶＬＣ）、ハフマン符号化、算術符号化、レンジ符号化のうちの少なくとも１つを含む。

【００２６】

オプションで、エンコーダ（１０）は、相互に類似のビットのランレングスに応じて入力データ（Ｄ１）を複数のセクションに分割するように動作可能であり、上記ランレングスは、ランレングス符号化（ＲＬＥ）、分割ＲＬＥ、エントロピー変更（ＥＭ）、ハフマン符号化、可変長符号化（ＶＬＥ）、レンジ符号化及び／又は算術符号化を用いる符号化に有効なものである。

【００２７】

オプションで、上記エンコーダにおいて、上記処理装置は、コンピュータハードウェアを用いて実施するものであり、該コンピュータハードウェアは、非一時的機械可読データ記憶媒体に記録された１つ又は複数のソフトウェア製品を実行するように動作可能である。

【００２８】

第２の態様によれば、数値列を含む入力データ（Ｄ１）を符号化するエンコーダを用いることにより、対応符号化入力データ（Ｄ２又はＤ３）を生成する方法であって、該方法は、

（ａ）差分符号化及び／又は合算符号化の方式を入力データ（Ｄ１）に適用することにより、１つ又は複数の対応符号化列を生成するエンコーダのデータ処理装置を用いること；並びに

（ｂ）符号化出力データ（Ｄ２又はＤ３）を生成するために、上記１つ又は複数の対応符号化列に対して最大値におけるラップアラウンド及び／又は最小値におけるラップアラウンドを行う、上記データ処理装置を用いること

を含み、

上記方法は、符号化出力データ（Ｄ２又はＤ３）を作成するのに利用される一連の予測値としてデフォルト第１予測値を利用することを含み、上記符号化データは、入力値、予測値及び符号化演算を用いて生成されることを特徴とする方法が提供される。

【００２９】

上記ラップアラウンドは、上記符号化法を実施する場合に、負の差分や極端に大きい値を生じさせないために利用されるという利点を持つ。「数値」という用語は、個々のビット（２進法）、並びにビットの群（２進数より高次）を含むものと解される。

【００３０】

オプションで、入力データ（Ｄ１）及び／又は上記１つ又は複数の対応符号化列を分析することにより、符号化出力データ（Ｄ２又はＤ３）の生成に用いるための上記１つ又は複数の対応符号化列に適用する１つ又は複数のオフセット値、最小値、及び／又は最大値を計算する上記データ処理装置を用いる。さらにオプションで、上記方法は、１つ又は複数のオフセット値に関し値「０」を用いることを含む。

【００３１】

オプションで、上記方法は、１つ又は複数の１ビット値を含む数値を処理すること、並びにビット単位方式で入力データ（Ｄ１）を符号化することを含む。さらにオプションで、上記方法は、１つ又は複数のオフセット値に関し値「０」を用いることを含む。

【００３２】

オプションで、上記方法を実施する際には、上記１つ又は複数の対応符号化列は、入力データ（Ｄ１）の連続値における変化を表す。

【００３３】

オプションで、上記方法は、ラップアラウンド値（ｗｒａｐＶａｌｕｅ）を利用するようにエンコーダを動作させることを含み、該ラップアラウンド値は、最大値（ｈｉｇｈＶａｌｕｅ）－最小値（ｌｏｗＶａｌｕｅ）＋１である。このようなラップアラウンドは、符号化出力データ（Ｄ２又はＤ３）を生成する際に、最小値（ｌｏｗＶａｌｕｅ）未満の値、しばしば負の値、又は最大値（ｈｉｇｈＶａｌｕｅ）を超える値を生じさせないために、有利に用いられる。

【００３４】

オプションで、上記方法は、別々に符号化される複数のデータセクションに入力データ

（Ｄ１）を分割することを含む。さらにオプションで、上記方法は、上記データセクションに対して選択的に符号化を適用することを含み、該符号化を適用することは、これによってデータ圧縮が符号化出力データ（Ｄ２又はＤ３）において達成可能である場合にのみ実行されるものである。

【００３５】

さらにオプションで、上記方法において、デフォルト第１予測値は「０」である。さらにオプションで、上記第１予測値は、また、最大値（ｈｉｇｈＶａｌｕｅ）、最小値（ｌｏｗＶａｌｕｅ）、（最大値（ｈｉｇｈＶａｌｕｅ）＋最小値（ｌｏｗＶａｌｕｅ）÷２等であり得る。

【００３６】

オプションで、上記方法は、更なる符号化を適用することにより、符号化出力データ（Ｄ２）を生成することを含み、該更なる符号化が、ランレングス符号化（ＲＬＥ）、分割ＲＬＥ、エントロピー変更（ＥＭ）、可変長符号化（ＶＬＣ）、ハフマン符号化、算術符号化、レンジ符号化のうちの少なくとも１つを含む。

【００３７】

さらにオプションで、上記方法は、相互に類似のビットのランレングスに応じて入力データを複数のセクションに分割することを含み、上記ランレングスは、ランレングス符号化（ＲＬＥ）、分割ＲＬＥ、エントロピー変更（ＥＭ）、ハフマン符号化、可変長符号化（ＶＬＥ）、レンジ符号化及び／又は算術符号化を用いる符号化に有効なものである。

【００３８】

さらにオプションで、上記方法は、コンピュータハードウェアを用いて上記処理装置を実施することを含み、該コンピュータハードウェアは、非一時的機械可読データ記憶媒体に記録された１つ又は複数のソフトウェア製品を実行するように動作可能である。

【００３９】

第３の態様によれば、符号化データ（Ｄ２、Ｄ３又はＤ４）を復号することにより対応復号出力データ（Ｄ５）を生成するデコーダであって、デコーダは、符号化データ（Ｄ２、Ｄ３又はＤ４）の１つ又は複数の部分を処理するデータ処理装置を備え、該データ処理装置は、上記１つ又は複数の部分の１つ又は複数の対応符号化列に対して差分復号及び／又は合算復号の方式を適用するように動作可能であり、上記１つ又は複数の符号化列は、復号出力データ（Ｄ５）を生成するために、最大値におけるラップアラウンド及び／又は最小値におけるラップアラウンドを受け、上記データ処理装置は、該装置を介して復号される一連のデータにおいて第１予測値のデフォルト値を仮定するように動作可能であることを特徴とする、デコーダが提供される。

【００４０】

オプションで、上記デコーダは、１つ又は複数の１ビット値を含む符号化データ（Ｄ２、Ｄ３又はＤ４）を復号するように動作可能であり、かつ上記デコーダは、ビット単位方式で符号化データ（Ｄ２、Ｄ３又はＤ４）を復号するように動作可能である。

【００４１】

オプションで、上記デコーダは、上記データ処理装置が、復号出力データ（Ｄ５）の生成に用いるための上記１つ又は複数の符号化列に適用する１つ又は複数の値を受信するように動作可能となるように構成されている。さらにオプションで、受信される値は、最大値、最小値、及び／又はオフセット値である。さらにオプションで、上記デコーダは、値「０」を有する１つ又は複数のオフセット値に関して機能するように動作可能である。

【００４２】

オプションで、上記デコーダは、符号化データ（Ｄ２、Ｄ３又はＤ４）に符号化される連続値における変化を表す１つ又は複数の対応符号化列に関して機能するように動作可能である。

【００４３】

オプションで、上記デコーダは、ラップアラウンド値（ｗｒａｐＶａｌｕｅ）を利用するように動作可能であり、該ラップアラウンド値は、最大値（ｈｉｇｈＶａｌｕｅ）－最

小値（ｌｏｗＶａｌｕｅ）＋１である。このようなラップアラウンドは、復号出力データ（Ｄ５）を生成する際に、最小値（ｌｏｗＶａｌｕｅ）未満の値、しばしば負の値、又は最大値（ｈｉｇｈＶａｌｕｅ）を超える値を生じさせないために用いられる。

【００４４】

オプションで、上記デコーダは、上記データ処理装置が、該装置を介して処理されるデータに対して、ランレングス符号化（ＲＬＥ）、分割ＲＬＥ、エントロピー変更（ＥＭ）、可変長符号化（ＶＬＣ）、ハフマン符号化、算術符号化、レンジ符号化の少なくとも１つの逆を適用するように動作可能となるように動作可能である。

【００４５】

さらにオプションで、上記デフォルト予測値は、値「０」を有する。しかしながら、上述のとおり、予測値は、また、最小値（ｌｏｗＶａｌｕｅ）、最大値（ｈｉｇｈＶａｌｕｅ）、（最大値（ｈｉｇｈＶａｌｕｅ）＋最小値（ｌｏｗＶａｌｕｅ）＋１）÷２等であり得る。

【００４６】

オプションで、上記デコーダにおいて、上記処理装置は、コンピュータハードウェアを用いて実装されるものであり、該コンピュータハードウェアは、非一時的機械可読データ記憶媒体に記録された１つ又は複数のソフトウェア製品を実行するように動作可能である。

【００４７】

第４の態様によれば、符号化データ（Ｄ２、Ｄ３又はＤ４）を復号するデコーダを用いることにより対応復号出力データ（Ｄ５）を生成する方法であって、該方法は、

上記復号データ（Ｄ２、Ｄ３又はＤ４）の１つ又は複数の部分を処理するデータ処理装置を用いることを含み、

上記データ処理装置は、上記１つ又は複数の部分の１つ又は複数の対応符号化列に対して差分復号及び／又は合算復号の方式を適用するように動作可能であり、上記１つ又は複数の符号化列が、復号出力データ（Ｄ５）を生成するために、最大値におけるラップアラウンド及び／又は最小値におけるラップアラウンドを受け、

上記データ処理装置は、該装置を介して復号される一連のデータにおいて第１予測値のデフォルト値を仮定するように動作可能であることを特徴とする方法が提供される。

【００４８】

オプションで、上記方法は、１つ又は複数の１ビット値を含む符号化データ（Ｄ２、Ｄ３又はＤ４）を復号すること、並びにビット単位方式で符号化データ（Ｄ２、Ｄ３又はＤ４）を復号するデコーダを用いることを含む。さらにオプションで、上記方法において、１つ又は複数のオフセット値は値「０」を有する。

【００４９】

オプションで、上記方法は、入力データ（Ｄ１）の値、及び／又は復号出力データ（Ｄ５）の生成に用いるための１つ又は複数の対応復号列に適用する１つ又は複数の対応符号化列（Ｄ２、Ｄ３又はＤ４）を受信するデータ処理装置を用いることを含む。

【００５０】

オプションで、上記方法において、上記１つ又は複数の対応符号化列は、符号化データ（Ｄ２、Ｄ３又はＤ４）に符号化される連続値における変化を表す。

【００５１】

オプションで、上記方法は、復号出力データ（Ｄ５）を生成する際に、最大値におけるラップアラウンド又は最小値におけるラップアラウンドを利用することにより、負の差分や極端に大きな値を生じさせないデータ処理装置を用いることを含む。

【００５２】

オプションで、上記方法において、上記データ処理装置は、該装置を介して処理されるデータに対して、ランレングス符号化（ＲＬＥ）、分割ＲＬＥ、エントロピー変更（ＥＭ）、可変長符号化（ＶＬＣ）、ハフマン符号化、算術符号化、レンジ符号化の少なくとも１つの逆を適用するように動作可能である。

【００５３】
さらにオプションで、上記方法において、上記デフォルト値は、値「０」を有する。
【００５４】
オプションで、上記方法において、上記処理装置は、コンピュータハードウェアを用いて実装され、該コンピュータハードウェアは、非一時的機械可読データ記憶媒体に記録された１つ又は複数のソフトウェア製品を実行するように動作可能である。
【００５５】
第５の態様によれば、符号化データ（Ｄ２、Ｄ３又はＤ４）を復号するデコーダを用いることにより対応復号出力データ（Ｄ５）を生成する方法であって、該方法は、
（ａ）上記符号化データ（Ｄ２、Ｄ３又はＤ４）を処理するものであり、上記符号化データ（Ｄ２、Ｄ３又はＤ４）は、変換データの連続値における変化を表す少なくとも１つの符号化列を含み、かつ最大値におけるラップアラウンド又は最小値におけるラップアラウンドを利用するものであることを考慮して、上記符号化データ（Ｄ２、Ｄ３又はＤ４）の１つ又は複数の部分に復号を適用するデータ処理装置を用いること、並びに
（ｂ）対応処理データを生成し、かつ少なくとも１つのプリオフセット値及び／又はポストオフセット値を用いて上記１つ又は複数の部分を変換することにより復号出力データ（Ｄ５）を生成するデータ処理装置を用いること、を含むことを特徴とする方法が提供される。
【００５６】
上述の態様に関連し、オフセット値については、オプションで、デコーダにおいて、上記データ処理装置は、符号化データ（Ｄ２、Ｄ３又はＤ４）における各種値を受信し、かつそれら値から少なくとも１つのプリオフセット値及び／又はポストオフセット値を導出するように動作可能である。上記ポストオフセット値は、逆演算、例えば、直接逆ＯＤｅｌｔａ演算を用いる前に変換データを生成するのに用いることが可能であり、上記プリオフセット値は、逆演算の後にデータを変換するのに用いられる。少なくとも１つのオフセット値は、オプションで、処理データと組合され、復号出力データ（Ｄ５）が生成される。
【００５７】
オフセットは、任意にデコーダにおいて使用可能である。オフセットは、それがエンコーダにおいてもまた用いられる際に用いられる。範囲（ｌｏｗＶａｌｕｅからｈｉｇｈＶａｌｕｅ）を用いるパラメータ（ｗｒａｐＶａｌｕｅ）内のラップアラウンド、逆演算（合算対差分）及び逆予測因子（入力値対出力値）は、デコーダの重要な要素である。
【００５８】
オプションで、上記デコーダにおいて、上記データ処理装置は、該装置を介して処理されるデータに対して、ランレングス符号化（ＲＬＥ）、分割ＲＬＥ、エントロピー変更（ＥＭ）、可変長符号化（ＶＬＣ）、ハフマン符号化、算術符号化、レンジ符号化の少なくとも１つの逆を適用するように動作可能である。この処理は、データ（Ｄ３）からデータ（Ｄ４）を生成する目的のために実行される。
【００５９】
上記デコーダにおいて、上記データ処理装置は、逆復号、例えば逆ＯＤｅｌｔａ復号の方式で実施される場合、範囲（ｌｏｗＶａｌｕｅからｈｉｇｈＶａｌｕｅ）を用いるパラメータ（ｗｒａｐＶａｌｕｅ）内のラップアラウンドを利用するように動作可能である。
【００６０】
上記第６の態様によれば、入力データ（Ｄ１）を符号化することにより対応符号化データ（Ｄ２又はＤ３）を生成する、第１の態様によるエンコーダを少なくとも１つ、並びに符号化データ（Ｄ２、Ｄ３又はＤ４）を復号することにより対応復号データ（Ｄ５）を生成する、第３の態様によるデコーダを少なくとも１つ備えるコーデック装置が提供される。
【００６１】
第７の態様によれば、不揮発性（非一時的）機械可読データ記憶媒体に記録されたソフ

トウェア製品であって、該ソフトウェア製品は、第２の態様による、データを符号化する方法を実施するためにコンピュータハードウェアで実行可能である、ソフトウェア製品が提供される。

【００６２】

第８の態様によれば、不揮発性（非揮発性）機械可読データ保存媒体に記録されたソフトウェア製品であって、該ソフトウェア製品は、第４の態様による、データを符号化する方法を実施するためにコンピュータハードウェアで実行可能である、ソフトウェア製品が提供される。

【００６３】

本発明の特徴は、添付の特許請求の範囲に規定される発明の要旨を逸脱することなく、各種組合せにおいて組合せることが可能であることが理解される。

【００６４】

以下の図面を参照して、本開示の実施形態を例としてのみ以下に説明する。

【図面の簡単な説明】

【００６５】

【図１】図１は、本開示に従って機能するように実施されるエンコーダ及びデコーダを備えるコーデック装置の図である。

【図２】図２は、図１のエンコーダにおいて実行される、データを符号化する方法の工程の図である。

【図３】図３は、図１のデコーダにおいて実行される、データを復号する方法の工程の図である。

【詳細説明】

【００６６】

添付の図面において、下線を付した番号は、その下線を付した番号が位置する要素、又はその下線を付した番号が隣接する要素を表すのに用いられる。下線が付されていない番号は、要素にその下線が付されていない番号を結合させる線によって特定されるその要素に関する。番号に下線が付されておらず、関連する矢印が付随する場合、下線が付されていないその番号が、その矢印が指し示す全体要素を特定するのに用いられる。

【００６７】

本開示の実施形態を説明する際、表２に示す通り以下の頭字後及び定義が用いられる。

【００６８】

表２：頭字語及び定義

頭字語	説明
ADC	アナログ・デジタル変換器（Analog-to-digital converter）
Codec	デジタルデータに関する符号器及び対応復号器
DAC	デジタル・アナログ変換器（Digital-to-analog converter）
DB	ランダムアクセスメモリ（ＲＡＭ）又は読出専用メモリ（ＲＯＭ）におけるデータベース
DC	所与の画像のＤＣ要素、即ち画像の平均値、即ち平均輝度に対応し、画像の最低空間周波数要素を表す。
RLE	ランレングス符号化（Run-length encoding）
ROI	対象とする範囲（Region of interest）
ROM	読出専用メモリ（Read Only Memory）
VLC	可変長符号（Variable-length code）

【００６９】

概して、図１を参照すると、本開示は、エンコーダ１０及びそれに関連する操作方法に関連し、エンコーダ１０は、直接ＯＤｅｌｔａエンコーダとして実施されるという利点を持つ。さらに、本開示は、対応するデコーダ２０にも関連し、デコーダ２０は、逆ＯＤｅｌｔａデコーダとして実施されるという利点を持つ。該開示による実施形態では、有利には、上述した既知の差分符号化法のビット最適化バージョン、並びにその他データに関するレンジ最適化バージョンである直接ＯＤｅｌｔａ演算を利用する。ＯＤｅｌｔａ符号化が、可変長データワード、例えば８／１６／３２／６４ビットを利用し、かつ／又は元の値が１～６４ビットの範囲で表現される８／１６／３２／６４ビットのデータ要素の可変長符号化を利用するコンピュータハードウェア又は専用デジタルハードウェアにおいて利用され、対応する符号化値が１～６４ビットで生成される。勿論、エンコーダ１０及びデコーダ２０は、如何なる場合においても、データＤ１、例えば元データにどの種類の数値が含まれているのかを認識し、従って、その定義又は伝送が、ここでさらに明らかになることはない。数値範囲（ＭＩＮ及びＭＡＸ）が既知であること、並びにデータＤ１が利用され得ることが単に仮定される。

【００７０】

既知の差分符号化法は、元（ＭＩＮ～ＭＡＸ）から結果（ＭＩＮ－ＭＡＸ～ＭＡＸ－ＭＩＮ）まで値の範囲を増加させる。この事は、該符号化法は、元データが正の値のみを含む場合、負の値もまた作成することを意味する。本開示によるＯＤｅｌｔａ演算は、対応する元の値の範囲にない値を作成することは決してなく、従って、使用されるデータ範囲を増加させることもなく、それ故に例えばエントロピー低減及び関連データ圧縮を実行する際に有利に利用される。例えば、既知の差分符号化法は、５ビット、即ち０～３１の値の範囲のデータストリームを用いて作動し、その結果、該差分符号化法によって生成されるデータ値は、－３１～＋３１の範囲、即ち６ビット（即ち符号ビット＋５ビット）を用いて実質的に表現され得る６３個の値になる。対照的に、直接ＯＤｅｌｔａ生成値は、上記の５ビットデータストリームから生成される際には、０～３１の範囲に依然としてある。さらに、既知の差分符号化法は、再起的に実施することは不可能であるが、本開示による直接又は逆ＯＤｅｌｔａ演算は、再帰的に実施可能であるにも関わらず、用いた値の範囲を依然として保存する。この値の範囲は、ビットに対して忠実である必要はなく、例え

ば、０～３１の値は５ビットで定義されるが、ＯＤｅｌｔａ演算は、任意の値の範囲、例えば０～２５の値の範囲を用いることができ、依然として適切に動作する。

【００７１】

原則として、本明細書に記載されるＯＤｅｌｔａ法は、常に、既存のデータ範囲に基づいて直接機能することが可能であり、そのデータ範囲の例は以下に挙げられる。ＯＤｅｌｔａ法は、データにおいて生じる最低値（「ｌｏｗＶａｌｕｅ」）及びデータにおいて生じる最高値（「ｈｉｇｈＶａｌｕｅ」）を示す情報を伝達することによって強化することもできる。ｌｏｗＶａｌｕｅ≧ＭＩＮかつｈｉｇｈＶａｌｕｅ≦ＭＡＸであること、並びにこれらの値は任意選択のものであることに留意されたい。

【００７２】

本開示による直接ＯＤｅｌｔａ演算及び逆ＯＤｅｌｔａ演算の２つの例が以下に記載されている。直接ＯＤｅｌｔａ演算及び逆ＯＤｅｌｔａ演算の最初の例は、例えば不揮発性（非一時的）機械可読データ記憶媒体に記録された１つ又は複数のソフトウェア製品を実行するように動作可能である電子ハードウェア及び／又はコンピュータハードウェアにおいて実施するのに、効率的であり、かつ比較的単純である。

【００７３】

本開示による直接ＯＤｅｌｔａ演算又は逆ＯＤｅｌｔａ演算を実施する際には、有利には、元のデータ値列は全て正であり、最低値は０である。オプションで、何らかのオフセット値、即ちプリオフセット値又はポストオフセット値が、データ値のすべてが正となり、かつ最低値が「０」となるように、それらデータ値をシフトさせるのに利用され得る。本開示によるＯＤｅｌｔａ演算は、直接方式で全種類のデータと共に容易に利用され得る。本開示によるＯＤｅｌｔａ演算は、典型的には、データ圧縮を提供することが可能であり、即ち伝達データレートを減らすことができる。その理由は、オフセット値がすべての値に加算されるか、或はすべての値から引かれる際に、データ値の範囲は、より少ないビットで定義され得るからである。例えば、直接ＯＤｅｌｔａ演算又は逆ＯＤｅｌｔａ演算を適用する前の元データ値は、－１１～＋１８の範囲にあり、この範囲は、＋１１のオフセット値を用いて０～２９の範囲に変換することができ、その変換範囲はその後、５ビットで記述される。このようなプリオフセット値又はポストオフセット値が利用されない場合、元データ値は、それらを記述するのに少なくとも６ビットを必要とし、しばしば、実際には、完全な８ビット符号付バイトが便宜上利用される。

【００７４】

データ範囲に対する同様の最適化が、一般化された直接ＯＤｅｌｔａ演算又は逆ＯＤｅｌｔａ演算を用いる際にもまた可能である。したがって、直接ＯＤｅｌｔａ演算若しくは逆ＯＤｅｌｔａ演算又は何らかのその他方法が、値の完全な範囲よりも小さいオフセット値で提示され得るデータ値を作成する場合、その範囲最適化は、ＯＤｅｌｔａ符号化法のどの段階においても実施され得る。オフセット値が、符号が負であろうと又は正であろうと用いられる際には、以下に図１、図２及び図３を参照して説明するとおり、オフセット値は、また、エンコーダ１０からデコーダ２０に伝送されなければならない。

【００７５】

直接ＯＤｅｌｔａ演算は、例えばビット単位方式で元データＤ１を符号化するために、１ビット方式で容易に実施され得る。このような１ビット方式で、以下により詳細に説明されるとおり方法１及び方法３により、図１の元データＤ１においてビット値の変化がない場合には値「０」が作成され、元データＤ１においてビット値に変化が生じた場合には値「１」が作成される。元データの最初のビットに関する予測は、任意に値「０」であり、従って、元データＤ１における最初のビット値が保存される。或は、元データの最初のビット予測値を値「１」として利用することも任意に可能であるが、このような選択は、符号化において如何なる利益も与えることはない。そのため、その予測が常に１ビットデータに対してデフォルトで値「０」であると仮定される場合、如何なる選択も伝送される必要はない。即ち、所定の値「０」が、エンコーダ１０及びデコーダ２０によって利用されることにより、この予測を伝達する必要がなくなり、その結果データ圧縮の向上に繋が

る。

【００７６】

本開示による直接ＯＤｅｌｔａ符号化の例を以下に記載する。例示である元ビット列、即ち１７個の「１」と２０個の「０」とを含む３７個のビットが、以下のとおり式１で与えられる。

【００７７】

［式１］

０１０１０１１００１０００１０１０００００００００００００１１１１１１１１１１

【００７８】

式１のエントロピーＥは、式２から計算可能である。

【００７９】

［式２］

$$E = 17 * \log_{10}\left(\frac{37}{17}\right) + 20 * \left(\frac{37}{20}\right) = 11.08523$$

【００８０】

式２においてエントロピーを符号化するのに必要なビットの数、即ちMin_bitは、上述の文書Ｄ７及びＤ８に記載されるとおり、つまり式３で与えられるとおり、シャノンの情報源符号化定理から計算可能である。

【００８１】

［式３］

$$\mathrm{Min\_bits} = \frac{\mathrm{E}}{\log_{10}(2)} = 36.82 \quad (\text{ビット})$$

【００８２】

元ビット列が、上述のとおり直接ＯＤｅｌｔａ演算、即ち方法１及び方法３で処理されると、１３個の「１」と２４個の「０」が存在する、３７個のビットを含む以下のビット列が生成される。

【００８３】

［式４］

０１１１１０１０１１００１１１１０００００００００００１０００００００００

【００８４】

式４のエントロピーＥは、式５から計算可能である。

【００８５】

［式５］

$$\mathrm{E} = 13 * \log_{10}\left(\frac{37}{13}\right) + 20 * \left(\frac{37}{24}\right) = 10.41713$$

【００８６】

式５は、ビット最小値、即ち、式６によるMin_bitsで表現可能である。

【００８７】
［式６］

$$\mathrm{Min\_bits} = \frac{E}{\log_{10}(2)} = 34.60 \quad (\text{ビット})$$

【００８８】
式４のビット列は、例えばランレングス符号化（ＲＬＥ）、ハフマン符号化、算術符号化、レンジ符号化、エントロピー変更器符号化、又はＳＲＬＥ符号化のうちの少なくとも１つを用いた更なる符号化で処理され、データ圧縮が達成されるという利点を持つ。
【００８９】
ＯＤｅｌｔａ演算は、その関連エントロピー符号化法が適用される場合、元データＤ１を表すのに必要なビット量を減らし、例えば、ＲＬＥ又はＳＲＬＥが、例えば式１にあるような元データの代わりに、例えば式４にあるような演算データに用いられ、この１ビット直接ＯＤｅｌｔａ演算は、即ち方法１及び方法３は、式１の元ビット列において多数の変化がある場合には複数の「１」を作成し、式１の元ビット列において相互に類似のビットの長いストリームがある場合には複数の「０」を生成する。
【００９０】
ＯＤｅｌｔａ演算の逆バージョン、即ち方法１及び方法３の逆は、符号化データストリーム、即ちデータＤ２に値「１」がある場合には、ビット値を、値「０」から値「１」に、又は必要に応じて値「１」から値「０」に変化させ、データＤ２の符号化ストリームに「０」の値がある場合にはビット値を変化させることはない。ＯＤｅｌｔａ演算が、直接ＯＤｅｌｔａ演算が為されたデータＤ２のビットストリームに対して実行される場合、データＤ１の元ストリームは、復号データＤ５として再生される。しかしながら、上述のとおり、ＶＬＣ又はハフマン符号化等の更なる符号化が有利に利用されるが、この事も考慮される必要がある。これは、データＤ３がエントロピーエンコーダの正方向の演算を用いてデータＤ２から生成され、データＤ４がエントロピーデコーダの逆演算を用いてデータＤ３から生成されることを意味する。
【００９１】
データＤ１の元ストリームは、符号化をそれに適用する前に、２以上のセクションに分割されるという利点を持つ。このような分割は、データＤ１の元ストリームを符号化する際により良い最適化が利用され得る機会を提供する。例えば、データＤ１における可変列が、直接ＯＤｅｌｔａ符号化される場合、即ち方法１及び方法３を利用して符号化される場合により多くの「１」を生成する。これに対して、フラットな不変列、即ち「フラット」列では、例えば後続のＶＲＬ符号化又はハフマン符号化にとって望ましい「０」が、より多く作成されるので、データＤ１を、上述のとおり別々に符号化できる複数のセクションに分割することによって、データＤ１を構成するビットストリーム全体に対してエントロピーＥを低減することができる。したがって、このような分割が有利となる。
【００９２】
本発明による直接ＯＤｅｌｔａ符号化の例を、相互に別々に符号化される複数のセクションが利用される場合について以下に説明する。元の単一ビット列を含む第１セクションは、以下式７のとおり、全体で１６個のビット、即ち７個の「１」及び９個の「０」を含む。
【００９３】
［式７］
０１０１０１１００１０００１０１
【００９４】

ここで、Ｈ（Ｘ）＝４．７６２１かつＢ＝１５．８２であり、Ｈはエントロピーを示し、ＢはMax_bitを示す。式７の元ビット列が、直接ＯＤｅｌｔａ演算で処理される場合、対応変換ビット列が式８のように与えられる。

【００９５】

［式８］

０１１１１０１０１１００１１１

【００９６】

ここで、Ｈ（Ｘ）＝４．３１５８かつＢ＝１４．３４である。

【００９７】

元の単一ビット列を含む第２セクションは、以下式９に示すとおりのビットを含む。

【００９８】

［式９］

０００００００００００１１１１１１１１１１

【００９９】

ここで、Ｈ（Ｘ）＝６．３１１３かつＢ＝２０．９７である。式９の元ビット列が直接ＯＤｅｌｔａ演算で処理されると、対応変換ビット列は、式１０のとおり与えられる。

【０１００】

［式１０］

０００００００００００１０００００００００

【０１０１】

ここで、Ｈ（Ｘ）＝１．７４６０かつＢ＝５．８０である。これらの例では、上述したとおり、Ｈ（Ｘ）はエントロピーＥを表し、Ｂは、符号化に要するビット最小数を表す。

【０１０２】

この例における式７及び式１０による最良の圧縮は、両セクションが別々に直接ＯＤｅｌｔａ演算で処理される場合に達成される（即ち、１４．３４ビット＋５．８０ビット＝全体で２０．１４ビットに符号化）。これは、もともと必要とされていた３６．８２ビットよりも少ないビットを要し、即ち直接ＯＤｅｌｔａ演算ビットは３４．６０ビット、又は分割後に要したビット数（＝１５．８２ビット＋２０．９７ビット＝３６．７９ビット）を要する。有利には、データＤ１の元ビットストリームのセクションへの分割を、元データＤ１、及び変更データ（即ちデータＤ２に含まれるような変更データ）の対応エントロピーＨを１つずつ分析することによって自動的に実行する。

【０１０３】

データＤ１に複数の長いランセクションがある場合に、ビット値が配列に沿って急激に変化する十分に大きなデータ領域があることを前提として、オプションで、データＤ１の部分を符号化される新規セクションに単に分割することによって粗い方式で、データ圧縮が実施される。オプションで、データＤ１の幾つかのセクションは、例えば個々の異なるビットが比較的少ない、相互に類似のビットの長いランが存在する場合、直接ＯＤｅｌｔａ演算を利用することなく符号化される。このような場合、直接ＯＤｅｌｔａ演算は、データ圧縮目的には有意な利益をもたらさない。

【０１０４】

データＤ１をより小さいセクションに分割することは、符号化データＤ２にデータを付与する更なるオーバーヘッドを生成するという欠点を有する。このようなオーバーヘッドは、例えば、全ての新しいセクションに関連するデータビット量又はデータバイト量を示す情報を含む。しかしながら、少なくとも特定量のオーバーヘッドデータ値を伝達する必要があると常に認められ、従って、所与のデータが２つのデータセクションに分割された場合には、追加のオーバーヘッドデータ値が１つのみ存在することになる。

【０１０５】

後に復号され得る符号化ビットストリームを達成するためには、エントロピー符号化が、直接ＯＤｅｌｔａ演算、例えばＶＬＣ、ハフマン符号化、算術符号化、レンジ符号化、ＲＬＥ、ＳＲＬＥ、ＥＭ等の後に有利に実施される。実際のデータ符号化と比較して、算

出エントロピーＥ及び最小ビット見積値に基づいて最適化計算を実行する方がより簡単であり、かつ計算上より効率的である。このような順番で実行することにより、顕著な速度の最適化が可能となり、しばしば、符号化データＤ２において最適なデータ圧縮の結果が達成される。或は、元のビット、アルファベット、数字、バイト及びワードのデータ、即ちデータＤ１にあるものが、何らかの他の方法で最初に符号化されることにより、エントロピー最適化ビットストリームが生成され、その後、直接ＯＤｅｌｔａ演算を用いてエントロピー最適化ビットストリームを変更し、対応符号化データ、即ちデータＤ２を提供するように、エントロピー最適化を実行することも可能である。さらに、このＯＤｅｌｔａ演算データは、データＤ２から、さらに他の符号化方法を用いて符号化し、データＤ３を生成することもできる。

【０１０６】

一般化された直接ＯＤｅｌｔａ演算は、データＤ１において用いられる様々な値を記述するパラメータ、即ち、その様々な値を提示するのに必要とされるビットの値又は数を利用する。さらに、ＯＤｅｌｔａ演算は、正及び負のオフセット値、言い換えると正及び負の「ペデスタル（基礎）」値の使用を可能にする方法において利用される。例えば、データＤ１が７個のビットを用いて提示され場合であって、即ち使用可能な「０」～「１２７」の値を有するが、「６０」～「１１５」の範囲の値のみを含む場合において、－６０のオフセット値がデータＤ１に適用されると、これによって、６個のビットのみを含む値として表すこともできる、「０」～「５５」の範囲の値を有する変換データが生成され、つまりある程度のデータ圧縮が、これによって達成可能となる。従って、この一般化直接ＯＤｅｌｔａ演算は、全範囲のデータ値が、即ち７個のビットにおいて表され、かつ８ビットバイトによって従来表されるデータ値がデータＤ１に存在する場合に、結果を向上させる。

【０１０７】

本発明によれば、直接ＯＤｅｌｔａ値、即ち方法１は、正の値（ｌｏｗＶａｌｕｅ＝ＭＩＮ＝０かつｈｉｇｈＶａｌｕｅ＝ＭＡＸ＝１２７、ｗｒａｐＶａｌｕｅ＝１２７－０＋１＝１２８）のみを有するデータに関する、以下の例示ソフトウェア符号の抜粋によって記述される手順を用いて容易に計算可能である。

【０１０８】

```
wrapValue = power(2, bits) = power(2, 7) = 128
prediction Value（予測値） = (lowValue + highValue + 1) div 2 = (wrapValue + 1)
div 2 + lowValue = 64
for all pixels（全てのピクセルについて）

begin
    if(originalValue >= predictionValue) then
        ODeltaValue = originalValue - predictionValue
    else
        ODeltaValue = wrapValue + originalValue - predictionValue
    predictionValue = originalValue
End
```

【０１０９】

さらに上記ＯＤｅｌｔａ演算を説明するために一例を以下に提供する。値の元配列は、式１１のとおりである。

【０１１０】

［式１１］

```
65, 80, 126, 1, 62, 45, 89, 54, 66
```

【０１１１】

対応する従来の差分符号化値は式１２のとおりである。

【０１１２】

［式１２］
65, 15, 46, -125, 61, -17, 44, -35, 12
【０１１３】
対応直接ＯＤｅｌｔａ符号化値は式１３のとおりである。
【０１１４】
［式１３］
1, 15, 46, 3, 61, 111, 44, 93, 12
【０１１５】
ここで、パラメータwrapValue内のラップアラウンドが利用される。
【０１１６】
逆ＯＤｅｌｔａ演算、即ち方法１が、逆ＯＤｅｌｔａ値を生成するために使用可能であり、例えば以下の例示ソフトウェアコードによって実施される。
【０１１７】
```
wrapValue = power(2, bits) = power(2, 7) = 128
predictionValue（予測値） = (wrapValue + 1) div 2 + lowValue = 64
for all pixels（全てのピクセルについて）

begin
    ODeltaValue = originalValue + predictionValue
    if (ODeltaValue >= wrapValue) then
        ODeltaValue = ODeltaValue - wrapValue
    predictionValue = ODeltaValue
end
```
【０１１８】
このソフトウェアコードが、式１３に対して実行、適用された場合、式１４で与えられる値が生成される。
【０１１９】
［式１４］
65, 80, 126, 1, 62, 45, 89, 54, 66
【０１２０】
本例では、ｗｒａｐＶａｌｕｅが２の冪乗の値として用いられる。これは必須ではなく、負の値がまた利用可能であるか、所与のデータ列においてプリオフセットによって範囲が変更される場合、ｗｒａｐＶａｌｕｅが、データの最高値より大きい任意の値、又は用いられる範囲よりも大きい値であってもよい。この特徴を示す更なる例を以下に示す。
【０１２１】
図１を参照した上述の説明を要約すると、本開示は、エンコーダ１０及びデコーダ２０に関連するものである。オプションで、エンコーダ１０及びデコーダ２０は、３０によって全体が示されているコーデック装置と組合せて利用される。エンコーダ１０は、元入力データＤ１を受信するように動作可能であり、元入力データＤ１は、例えば直接ＯＤｅｌｔａ法を用いて符号化されることにより、対応符号化データＤ２又はＤ３が生成される。符号化データＤ２又はＤ３は、オプションで、通信ネットワーク４０を介して伝達されるか、又はデータ記憶媒体５０、例えば光ディスク読取専用メモリ（ＲＯＭ）等のデータ媒体に記憶される。デコーダ２０は、例えば通信ネットワーク４０を介してストリーム配信される符号化データＤ２又はＤ３、又はデータ記憶媒体５０に提供される符号化データＤ２又はＤ３を受信し、かつ逆の方法、例えば逆ＯＤｅｌｔａ法を適用することにより、例えば元データＤ１に実質的に類似する対応復号データＤ５を生成するように動作可能である。エンコーダ１０及びデコーダ２０は、例えば本明細書における例示実施形態として提供されるコードのように、１つ又は複数のソフトウェア製品を実行するように動作可能である、デジタルハードウェア、例えばコンピュータハードウェアを用いて実施されるという利点を持つ。或は、エンコーダ１０及び／又はデコーダ２０は、専用デジタルハードウ

ェアを用いて実施される。

【０１２２】

エンコーダ１０において実行されるＯＤｅｌｔａ法は、図２に記載される工程を利用するものである。任意の第１工程１００において、入力データＤ１が処理され、そのデータ要素の値の範囲が見出される。任意の第２工程１１０において、データ要素を正のレジームに変換することにより変換要素の対応セットを生成するために、その値の範囲から、オフセット、即ちプリオフセットが計算される。第３工程１２０においては、第２工程１１０において任意に変換された要素が、次いで、直接ＯＤｅｌｔａ符号化を受けることにより、対応ＯＤｅｌｔａ符号化値が生成される。第４工程１３０においては、ＯＤｅｌｔａ符号化値、並びに任意のオフセット値、最小値（ｌｏｗＶａｌｕｅ）、及び／又は最大値（ｈｉｇｈＶａｌｕｅ）が、次いで、例えばランレングス符号化（ＲＬＥ）、レンジ符号化、又はハフマン符号化を用いて別々に符号化され、データＤ２からデータＤ３が生成される。オフセット値、最小値（ｌｏｗＶａｌｕｅ）、及び／又は最大値（ｈｉｇｈＶａｌｕｅ）は、常に圧縮可能であるという訳ではなく、従って、それらはエンコーダ１０からデコーダ２０に適切なビット量を用いて伝達されることを要する。さらに、オフセット値、最小値（ｌｏｗＶａｌｕｅ）、及び／又は最大値（ｈｉｇｈＶａｌｕｅ）は、直接ＯＤｅｌｔａ演算に関する任意の特徴であり、例えば、オフセット値は、特定の状況では、値「０」を有し、ｌｏｗＶａｌｕｅは、値ＭＩＮを有し、ｈｉｇｈＶａｌｕｅは、値ＭＡＸを有する。即ち、変換は全く適用されずに、全範囲が用いられる。特に、直接ＯＤｅｌｔａ演算が、１ビットデータ、即ちビット単位の符号化のために実施される場合、オフセット値を必要とすることは全くなく、そのため、工程１００及び１１０は常に無視される。オフセット値が、工程１１０においてもまた用いられる場合には、最高値と最低値を提示する範囲値は、その中で更新されなければならない。異なる値の数、即ちｗｒａｐＶａｌｕｅは、デコーダ２０によってまた識別されなければならず、或は、さもなければエンコーダ１０は、圧縮データ内においてそれをデコーダ２０に伝達しなければならない。オプションで、デフォルトｗｒａｐＶａｌｕｅ（＝ｈｉｇｈＶａｌｕｅ－ｌｏｗＶａｌｕｅ＋１）が、エンコーダ及びデコーダにおいて用いられる。オプションで、エンコーダ１０及びデコーダ２０の少なくとも１つが、例えば入力データＤ１を符号化のためのセクションに分割してデータＤ１の最適な圧縮を提供し、符号化データＤ２を生成する最適な方式を見出すために、再帰的な方式で動作する。

【０１２３】

デコーダ２０において実行される逆ＯＤｅｌｔａ法は、図３に記載される工程を利用する。第１工程２００において、データＤ２／Ｄ３又はＤ４は、上述の工程１３０において利用されるものとは逆の符号化を受けることにより、復号ＯＤｅｌｔａデータが生成され、復号ＯＤｅｌｔａデータは、ＯＤｅｌｔａ符号化値を有し、かつ任意の別のオフセット値を有する。第２工程２１０においては、ＯＤｅｌｔａ符号化値は、復号され、データ要素列が生成される。第３工程２２０においては、データ要素列が、最適プリオフセット値を用いて変換され、復号データＤ５が生成される。特定の状況においては、このような変換は値「０」に設定され、つまり、有効な変換が適用されることはない。この場合も、例えば１ビット符号化、即ちビット単位の符号化を実行する場合、オフセット値を利用する必要なく、上記方法を実行することが可能であり、それによって、工程２２０を無視することが可能になる。さらに、デコーダ２０は、受信したデータ要素を適切な方式で復号することを可能にするためにｗｒａｐＶａｌｕｅを識別することも要する。

【０１２４】

上記オフセットを利用し、正の値のみを取得することによって、データＤ２又はＤ３におけるより効率的なデータ圧縮が達成可能となる。すべてのデータ値が、既に正の値である場合、如何なるオフセット値を追加する必要はない。勿論、以下の例に示すとおり、負のオフセット値は、利用可能な範囲を小さくするために任意に利用されるが、必須ではない。

【０１２５】

図２及び図３の方法は、オプションで、ＯＤｅｌｔａ符号化を受ける利用可能な値のみを用いることによって更に最適化され得る。このような最適化には、用いる値が既知であることが必要とされる。例えば、上述の例においては、１（＝元の最小値）～１２６（＝元の最大値）の値のみが元データセットＤ１に存在する。このため、オフセット値は、１である（->ｌｏｗＶａｌｕｅ＝元の最小値－オフセット＝１－１＝０かつｈｉｇｈＶａｌｕｅ＝元の最大値－オフセット＝１２６－１＝１２５）。プリオフセット値が元データＤ１から引かれた場合、結果として式１５にある次の値が得られる。

【０１２６】

［式１５］

64, 79, 125, 0, 61, 44, 88, 53, 65

【０１２７】

式１５から、最大値として１２５が決定され（ｈｉｇｈＶａｌｕｅ＝元の最大値（ｏｒｉｇｉｎａｌ ｍａｘ）－オフセット＝１２６－１＝１２５）、その結果、「数字」（＝最大差分値（ｍａｘｉｍｕｍ Ｄｅｌｔａ ｖａｌｕｅ）＝ｈｉｇｈＶａｌｕｅ － ｌｏｗＶａｌｕｅ）は、そうすると、１２５となり、即ちｗｒａｐＶａｌｕｅは、最小で１２６となり得る（＝数字＋１＝ｈｉｇｈＶａｌｕｅ－ｌｏｗＶａｌｕｅ＋１）。すると、これらの値を保存するか、及び／又は伝達する必要があるため、前の例は、以下のとおりプロセス値を変化させることによって変更することができる。

【０１２８】

wrapValue = 126　（「０」～「１２５」=>１２６個の異なる値）

prediction Value（予測値）= (highValue + lowValue + 1) div 2 = (wrapValue + 1) div 2 + lowValue = 63

【０１２９】

対応する直接ＯＤｅｌｔａ演算の値は式１６で与えられている。

【０１３０】

［式１６］

1, 15, 46, 1, 61, 109, 44, 91, 12

【０１３１】

全ての「負の差分値」が、今や、２分の１に減じられている（即ち＝範囲変化＝１２８－１２６）ことが理解される。同様に、デコーダ２０においても、プロセス値は以下のとおり変化させなければならない。

【０１３２】

wrapValue = 126

predictionValue = (wrapValue + 1) div 2 + lowValue = 63

【０１３３】

対応する逆のＯＤｅｌｔａ値は、以下の式１７のとおりである。

【０１３４】

［式１７］

64, 79, 125, 0, 61, 44, 88, 53, 65

【０１３５】

プリオフセット値を式１７に加算すると、式１５における元データに対応して、以下の式１８の結果が得られる。

【０１３６】

［式１８］

65, 80, 126, 1, 62, 45, 89, 54, 66

【０１３７】

本例では、値の範囲はほぼ全体であり、従って、オフセット値及び最大値（ｈｉｇｈＶａｌｕｅ）を用いて直接ＯＤｅｌｔａ演算を適用することにより得られる利益は比較的それほど大きいものではない。しかしながら、それら値が適切に伝達される場合には、エントロピーＥの低減が依然として達成可能であり、即ち、頻度テーブル又は符号テーブルに

おいて値の数を減らすことができる。この範囲がより小さい場合に最大の利益が得られる。

【０１３８】

データを符号化及び復号する実践的な１ビットの直接及び逆ＯＤｅｌｔａ法の例示実施形態、即ち、方法１又は方法３が、実行可能なコンピュータソフトウェアコードにより以下に提供される。これらの方法は、上記の直接及び逆ＯＤｅｌｔａ演算、即ち方法１又は方法３を利用する。ソフトウェアコードは、コンピュータハードウェアで実行される際に動作可能であり、１つのバイトバッファから別のバイトバッファに対してビットを処理する。ソフトウェアコードでは、ＧｅｔＢｉｔ、ＳｅｔＢｉｔ、及びＣｌｅａｒＢｉｔの関数が、常に、ＨｅａｄｅｒＢｉｔｓ値を更新する。ＨｅａｄｅｒＩｎｄｅｘ値もまた、次のビットが次のバイトにある場合に更新される。オプションで、ソフトウェアコードは、１セットのＨｅａｄｅｒＩｎｄｅｘ値及びＨｅａｄｅｒＢｉｔｓ値のみがソース及び宛先に関し用いられるように最適化することができ、その結果、所与のビットが宛先バッファに書き込まれる場合のみ値が更新される。

【０１３９】

```
procedure EncodeODelta1u(APtrSrc : PByte; ASrcDstBitLen : PCardinal; APtrDst : PByte)
var
 iSrcHeaderIndex, iSrcHeaderBits, iIndex,
 iDstHeaderIndex, iDstHeaderBits : Cardinal;
 bBit, bLastBit : Boolean;
begin
 // オフセットをリセット。
 iSrcHeaderIndex := 0;
 iSrcHeaderBits := 0;
 iDstHeaderIndex := 0;
 iDstHeaderBits := 0;

 // 差分値を初期化。
 bLastBit := False;

 // 全てのビットを処理。
 for iIndex := 0 to ASrcDstBitLen^-1 do
 begin
  // ビットを読み取る。
  bBit := GetBit(APtrSrc, @iSrcHeaderIndex, @iSrcHeaderBits);

  // 現ソースビットが前のソースビットと異なる場合には宛先ビットをセット。
  if (bBit <> bLastBit) then
  begin
   SetBit(APtrDst, @iDstHeaderIndex, @iDstHeaderBits);
   bLastBit := bBit;
  end
  else ClearBit(APtrDst, @iDstHeaderIndex, @iDstHeaderBits);
 end;
end;
```

```
function DecodeODelta1u(APtrSrc : PByte; ASrcDstBitLen : PCardinal; APtrDst : PByte) :
Boolean;
var
  iSrcHeaderIndex, iSrcHeaderBits, iIndex,
  iDstHeaderIndex, iDstHeaderBits : Cardinal;
  bBit, bLastBit : Boolean;
begin
  // オフセットをリセット。
  iSrcHeaderIndex := 0;
  iSrcHeaderBits := 0;
  iDstHeaderIndex := 0;
  iDstHeaderBits := 0;

  // 差分値を初期化。
  bLastBit := False;

  // 全てのビットを処理。
  for iIndex := 0 to ASrcDstBitLen^-1 do
  begin
   // ビットを読み取る。
   bBit := GetBit(APtrSrc, @iSrcHeaderIndex, @iSrcHeaderBits);

   // ソースビットが真(True)である場合にはビット値を変更。
   if (bBit = True) then
   begin
    if (bLastBit = True) then
     bLastBit := False
    else bLastBit := True;
   end;

   // ビット値(真(True)又は偽(False))に基づいて宛先ビットをセット。
   if (bLastBit) then
    SetBit(APtrDst, @iDstHeaderIndex, @iDstHeaderBits)
   else ClearBit(APtrDst, @iDstHeaderIndex, @iDstHeaderBits);
  end;
 end;
```

【０１４０】

上述の直接及び逆ＯＤｅｌｔａ演算、即ち方法１又は方法３は、例えばビデオデータ、画像データ、音声データ、グラフィックデータ、地震データ、医療データ、測定値、参照

数字及びマスク等のデジタルフォーマットである任意の種類のデータを圧縮するのに有利に利用される。さらに、１つ又は複数のアナログ信号も、対応デジタルデータに最初に変換させる場合に、例えば圧縮前にＡＤＣを用いることによって、直接ＯＤｅｌｔａ演算を用いて圧縮可能である。逆ＯＤｅｌｔａ演算を用いる際、データを変換して１つ又は複数のアナログ信号に戻すことが望まれる場合には、演算の後にＤＡＣを用いることができる。しかしながら、直接ＯＤｅｌｔａ演算それ自体は、データを圧縮するのに通常は効果的ではないものの、例えば可変長符号化（ＶＬＣ）、算術符号化、レンジ符号化、ランレングス符号化、ＳＲＬＥ、エントロピー変更器等の他の符号化法と組合せて利用される場合には効果的なデータ圧縮を提供できることが理解される。これらの符号化法は、直接ＯＤｅｌｔａ演算がエンコーダ１０において利用された後に、データＤ２に対して用いられる。生じるデータがデコーダ２０において実施される逆ＯＤｅｌｔａ演算に伝達される前に、符号化データＤ２は、対応的に復号されて元に戻されなければならない。ＯＤｅｌｔａ演算は、他の種類のエントロピー変更器と共に利用することもできる。特定の状況においては、直接ＯＤｅｌｔａ演算は、エントロピーＥの増加を生じさせることもあり、データ圧縮アルゴリズムは、データの符号化に用いるための直接ＯＤｅｌｔａ演算を、それが有利なデータ圧縮性能を提供する場合のみ、選択的に利用するように有利に動作可能であり、例えば、それは圧縮されるデータの性質に基づいて選択的に利用されたり、例えば上述のとおり入力データＤ１の選択部分に選択的に利用されたりする。

【０１４１】

直接ＯＤｅｌｔａ演算は、例えば、本願において援用される米国特許出願ＵＳ１３／５８４，００５に記載されるようなブロックエンコーダと組合せて利用されること等が考案されている。逆ＯＤｅｌｔａ演算は、本願において援用される米国特許出願ＵＳ１３／５８４，０４７に記載されるようなブロックデコーダと組合せて利用されることが考案されている。オプションで、直接ＯＤｅｌｔａ演算及び逆ＯＤｅｌｔａ演算は、本願において援用される米国特許出願ＵＳ１３／６５７，３８２に記載されるようなマルチレベル符号化法と組合せて利用されるという利点を持つ。２進状態を含む全ての種類の１ビットデータ、例えばデータＤ１に存在するもの等を１ビットバージョンの直接ＯＤｅｌｔａ演算で処理されることにより対応変換データを生成し、対応変換データは、次いで、その後、実際のエントロピー符号化を受けることにより、符号化データＤ２又はＤ３が生成されるという利点を持つ。オプションで、上述のとおり、直接ＯＤｅｌｔａ演算が、元データＤ１の性質に応じて選択的に利用される。

【０１４２】

オプションで、直接ＯＤｅｌｔａ演算の前又は後に、データのエントロピーを変更する他の方法を利用することができる。例えば、一般化されたバージョンの直接ＯＤｅｌｔａ演算の中でマルチビットデータに対して直接、直接ＯＤｅｌｔａ演算を用いることもできる。さらに、用いられるビット全てをビットの連続配列に最初に入力した後に、上記１ビットバージョンの直接ＯＤｅｌｔａ演算をマルチビットデータに対して有利に利用する。

【０１４３】

エンコーダ１０において直接ＯＤｅｌｔａ演算と組合せて複数の方法がデータ圧縮のために利用される場合、対応する逆演算が、例えば以下のとおり、デコーダ２０において逆順に実行される。

【０１４４】

以下の方法の順序がエンコーダ１０において利用される。

【０１４５】

［式１９］

［データＤ１］　　＝＞直接ＯＤｅｌｔａ（方法２）
＝＞ＶＬＣ－
＝＞ＥＭ
＝＞算術符号化
＝＞　［データＤ３］

【０１４６】
以下の方法の逆順がデコーダ２０において利用される。
【０１４７】
［式２０］
［データＤ３］　　＝＞逆算術符号化
＝＞逆ＥＭ
＝＞逆ＶＬＣ
＝＞逆ＯＤｅｌｔａ（方法２）
＝＞［データ Ｄ５］
【０１４８】
ここで、「ＶＬＣ」は可変長符号化を示し、「ＥＭ」はエントロピー変更を示す。
【０１４９】
上記に記載したＯＤｅｌｔａ演算は、可逆的であり、かつ無損失である。さらに、上記ＯＤｅｌｔａ演算は、オプションで、例えばビット単位の符号化を実行する際には１ビットデータストリームに対して、さらにそれだけではなくその他データに対しても、特に実施することが容易に可能である。すべての種類のデータが、一般化バージョンの直接ＯＤｅｌｔａ演算を用いて容易に処理され得るという利点を持つ。そして、直接ＯＤｅｌｔａ演算はデータが圧縮される際に利用され、対応する逆ＯＤｅｌｔａ演算は圧縮データを展開する際に利用されるという利点を持つ。オプションで、ＯＤｅｌｔａ演算が利用される場合、直接ＯＤｅｌｔａ演算と、それに対応する逆演算が、逆順に利用され、言い換えると、逆ＯＤｅｌｔａ演算は、元ビットストリームに対して最初に一時的に実行され、その後直接ＯＤｅｌｔａ演算が続き、元ビットストリームが再生される。一方のＯＤｅｌｔａ演算はエントロピーを増やし、他方のＯＤｅｌｔａ演算はエントロピーを減らす。直接ＯＤｅｌｔａ演算がエントロピーを全く変更してはならず、次いで逆ＯＤｅｌｔａ演算もエントロピーを変更しないというのは極めて稀有なケースである。例えば方法１のために直接ＯＤｅｌｔａ演算及び逆ＯＤｅｌｔａ演算が用いられる場合、これらの演算の逆順は、方法４の通常の順序と同様である。順序の同様の変更が、方法２及び方法３についても可能である。
【０１５０】
１ビットバージョンにおいて、即ちビット単位方式でデータを符号化するために、直接ＯＤｅｌｔａ演算は、予測無しで有利に開始し、即ち、デフォルトで初期値「０」の予測を仮定する。一般化バージョンにおいては、ＯＤｅｌｔａ演算は、使用可能なデータ範囲の半分を表す予測を用いて開始し、例えば、５ビットがデータＤ１の入力データ値に用いられる場合、即ち「０」～「３１」の範囲の３２個の異なる値が用いられる場合、予測値は３２/２＝１６である。有利には、ＯＤｅｌｔａ演算には、データ要素が該演算を用いて処理されるために、使用可能なデータ範囲に関する情報が提供される必要がある。
【０１５１】
上記に記載される発明の実施形態によれば、データＤ１においてビット又は任意のデジタル値として提示されるエントロピーＥを低減することが可能になる。直接ＯＤｅｌｔａ演算によって、ほぼ常に、差分符号化と比較してより向上したエントロピー低減を行える。差分符号化がバイトラップアラウンドと共に用いられ、元予測を用いた差異ＯＤｅｌｔａ演算（方法１）が、ｗｒａｐＶａｌｕｅ＝２５６、ｌｏｗＶａｌｕｅ＝ＭＩＮ＝０、及びｈｉｇｈＶａｌｕｅ＝ＭＡＸ＝２５５の値を用いる場合のみ、その中で同一の出力結果を生成する。別の直接ＯＤｅｌｔａ法が用いられる場合、又は入力データにおいて全データ範囲が利用可能ではない場合、ＯＤｅｌｔａ演算は、選択した方法、又はｌｏｗＶａｌｕｅ及び／若しくはｈｉｇｈＶａｌｕｅを送信することによってより良い結果を生成するが、即ち、これら送信されるものはｗｒａｐＶａｌｕｅもまた自動的に変更する。エントロピーが小さいほど、より高いデータ圧縮比でデータを圧縮することが可能になる。より高いデータ圧縮比は、より小さい容量のデータ記憶媒体の利用を可能にし、また、圧縮データを伝送する場合に、より遅いデータ帯域幅の利用も可能にし、対応するエネルギー消

費の低減も可能にする。

【０１５２】

上記において、差分と合算の方式の計算がエンコーダ１０において実行され、かつ対応する逆の計算がデコーダ２０において実行されることが理解される。エンコーダ１０において用いられる他の予測方法を用いることも可能であり、対応する逆予測が次いでデコーダ２０において実行される。この事は、実際、少なくとも４つの異なる直接ＯＤｅｌｔａ方法、並びに少なくとも４つの対応する逆ＯＤｅｌｔａ方法が存在することを意味している。これらの方法の詳細かつ厳密な説明は以下のとおりである。オプションで、上記計算は、再帰的な方式で実行されることにより、符号化データＤ２（又はＤ３）において高度なデータ圧縮が得られる。このような帰納的な計算を実行する際には、変化する数の範囲が、再帰的計算が何回利用されてきたかに応じて利用される。例えば、エンコーダ１０においては、以下の順序の計算がデータＤ１に対して実行され、符号化データＤ２（又はＤ３）が生成される。

【０１５３】

［式２１］

［データＤ１］　ｅ直接ＯＤｅｌｔａ（方法３）＝＞
ｅ直接ＯＤｅｌｔａ（方法３）＝＞
ｅＥＭ＝＞
ｅ直接ＯＤｅｌｔａ（方法１）＝＞
ｅＶＬＣ　［データＤ３］

【０１５４】

かつ、デコーダ２０において対応する逆演算が実行される。

【０１５５】

［式２２］

［データＤ３］　ｄＶＬＣ＝＞
ｄ逆ＯＤｅｌｔａ（方法１）＝＞
ｄＥＭ＝＞
ｄ逆ＯＤｅｌｔａ（方法３）＝＞
ｄ逆ＯＤｅｌｔａ（方法３）　［データＤ５］

【０１５６】

式２１（方法１に対応）、式２２（方法２に対応）、式２３（方法３に対応）、及び式２４（方法４に対応）によって示されるとおり、データがこれら４つの方法において演算処理されるたびに、任意にすべての方法を用いることを試すことが可能である。その理由は、これらの方法のうち１つが、その他の方法よりも、処理されるデータのエントロピーを減らせる可能性があるからである。エンコーダ１０及び／又はデコーダ２０内における方法の使用を最適化する上で、選択される１つの方法又は複数の方法が、必要とされるデータにおける情報量と比較してエントロピーを減少させる限り、何度も同一又は異なる方法を用いることは有利である。従って、方法１～４は、数値を符号化するために使用可能であるが、「数値」とは、その定義において、ビット単位方式の符号化ビットストリームにあるような１ビットデータ並びに非２進数、並びにマルチビット値を包含する。

【０１５７】

差分演算は、連続数値の余りを表し、それに対応して合算演算は、連続数値の合計を表す。エンコーダ１０において実行されるこれらの演算には、デコーダ２０におけるそれぞれ独自の対応する逆演算がある。差分又は合計は、現入力値、及び予測値として用いられる前の入力値又は結果値に基づいて計算され得る。他の予測値も用いることができ、それらは、例えば、デコーダにおいて逆を実行することが可能である限りにおいて、エンコーダにおいて以前の入力値及び出力値を用いることにより予測を作成し得る。

【０１５８】

このような方法の何れも、エンコーダ１０及びデコーダ２０内で有意にデータを圧縮するものではないが、すべての方法は、エントロピーを低減するのに有利に利用され、その

結果、他の圧縮法により、次いで、エントロピー低減データをより効率的に圧縮することができる。このような他の圧縮法は、オプションで、ハフマン符号化、算術符号化、レンジ符号化、ＲＬＥ符号化、ＳＲＬＥ符号化、エントロピー変更器符号化の少なくとも１つである。しかしながら、全ての方法に関し、例えばデータの可逆圧縮及びそれに続く可逆展開が達成し得る場合には、幾つかの数値を伝送する必要があり、その幾つかの数値を用いて上記演算とその逆演算とが常に正確に実施し得る。勿論、エンコーダ１０及びデコーダ２０は、どのような種類の数値が入力データＤ１に含まれているかに関する情報を有する。有利には、数値範囲、即ちＭＩＮ及びＭＡＸによって定義される数値範囲が認識されることが想定されている。原則、方法は、常に、直接既存のデータ範囲に基づいて機能し得る。上記演算が必要とする数値は、生じる最低値（ｌｏｗＶａｌｕｅ）及び生じる最高値（ｈｉｇｈＶａｌｕｅ）であり、ｌｏｗＶａｌｕｅはＭＩＮ以上であり、ｈｉｇｈＶａｌｕｅはＭＡＸ以下である。

【０１５９】

これらの値に基づいて、他の必要な数値を導くことができる。これらの値は、様々な形態で伝送され、欠損値が有利に計算されるという利点を持つ。例えば、セット［「ｌｏｗＶａｌｕｅ」、「ｈｉｇｈＶａｌｕｅ」、「ｎｕｍｂｅｒ（数）」］からの２つの値が既知である場合、その「ｎｕｍｂｅｒ（数）」は、［ｈｉｇｈＶａｌｕｅ －ｌｏｗＶａｌｕｅ］であり、次いで、第３の値は、それらから計算することができる。データＤ２において特定の値を省略し、次いでデコーダ２０においてそれら値を導くことは、データＤ２においてより大きな程度のデータ圧縮を提供することを可能にする。

【０１６０】

これらの値に加えて、第１の値、即ち「ｐｒｅｄｉｃｔｉｏｎ」（「予測」）の計算において前の値として用いられ得る数Ｐが必要とされる。「０」と「ｎｕｍｂｅｒ」（「数」）の値との間の値は、常に、数Ｐ、即ち「ｐｒｅｄｉｃｔｉｏｎ」（「予測」）に関して選択され得る。さらに、上記演算が、エンコーダ２０においてデータＤ２／Ｄ３又はＤ４を復号する際に回復可能に機能するためには、つまり、演算が生成する値の範囲をできる限り小さく縮小させるためには、値「ｗｒａｐＶａｌｕｅ」が上記演算に提供される必要がある。しかしながら、この「ｗｒａｐＶａｌｕｅ」は、「ｎｕｍｂｅｒ」（「数」）よりも大きくなければならず、有利には、それは値「ｎｕｍｂｅｒ」（「数」）＋１を有する。オプションで、例えばデータＤ１が大きい値よりも小さい値をより多く含むと仮定した場合、データＤ１の性質に応じて、第１「ｐｒｅｄｉｃｔｉｏｎ」（「予測」）値には、上記のとおり「０」が選択され得る。或は、データＤ１が小さい値よりも大きい値をより多く含むと仮定した場合、第１「ｐｒｅｄｉｃｔｉｏｎ」「予測」値には、「ｎｕｍｂｅｒ」「数」と等しい値が選択され得る。仮定が値の大きさに関して為されていない場合、「ｐｒｅｄｉｃｔｉｏｎ」（「予測」）値には値「（ｗｒａｐＶａｌｕｅ＋１）÷２＋ｌｏｗＶａｌｕｅ」を用いることが望ましい。

【０１６１】

本開示の実施形態を実施する際にコンピュータハードウェアにおいて実行する演算の例について以下に説明する。

【０１６２】

エンコーダ１０において、第１直接差分演算、即ち方法１は、有利には以下のとおり実施される。全てのデータ値に関し、入力値、即ち「ｏｒｉｇｉｎａｌ」値（「元」）値に対応する出力値、即ち「ｒｅｓｕｌｔ」（「結果」）がソフトウェアループにおいて計算される。

【０１６３】

```
  result = original - prediction
if result < lowValue then result = result + wrapValue
```

【０１６４】

最終的に、次の入力に関する予測値が現入力と等しい値に設定され、即ち、以下とされる。

【０１６５】
Prediction = original
【０１６６】
デコーダ２０において、第１逆差分演算、即ち、方法１が、有利には、以下のとおり実施される。全てのデータ値に関し、入力値、即ち「ｏｒｉｇｉｎａｌ」（「元」）値に対応する出力値、即ち「ｒｅｓｕｌｔ」（「結果」）がソフトウェアループにおいて計算される。
【０１６７】
result = original + prediction
if result > highValue then result = result - wrapValue
【０１６８】
最終的に、次の入力に関する予測値が、現結果と等しい値に設定され、即ち、以下のとおりとされる。
【０１６９】
prediction = result
【０１７０】
エンコーダ１０において、第２直接差分演算、即ち方法２が、有利には以下のとおり実施される。全てのデータ値に関し、入力値、即ち「ｏｒｉｇｉｎａｌ」（「元」）値に対応する出力値、即ち「ｒｅｓｕｌｔ」（「結果」）がソフトウェアループにおいて計算される。
【０１７１】
result = original - prediction
if result < lowValue then result = result + wrapValue
【０１７２】
最終的に、次の入力に関する予測値が、現結果と等しい値に設定され、即ち、以下のとおりとされる。
【０１７３】
prediction = result
【０１７４】
デコーダ２０において、第２逆差分演算、即ち方法２が、有利には以下のとおり実施される。全てのデータ値に関し、入力値、即ち「ｏｒｉｇｉｎａｌ」（「元」）値に対応する出力値、即ち「ｒｅｓｕｌｔ」（「結果」）がソフトウェアループにおいて計算される。
【０１７５】
result = original + prediction
if result > highValue then result = result - wrapValue
【０１７６】
最終的に、次の入力に関する予測値が、現入力と等しい値に設定され、即ち、以下のとおりとされる。
【０１７７】
prediction = original
【０１７８】
エンコーダ１０において、第１直接合算演算、即ち方法３が、有利には以下のとおり実施される。全てのデータ値に関し、入力値、即ち「ｏｒｉｇｉｎａｌ」（「元」）値に対応する入力値、即ち「ｒｅｓｕｌｔ」（「結果」）がソフトウェアループにおいて計算される。
【０１７９】
result = original + prediction
if result > highValue then result = result - wrapValue
【０１８０】

最終的に、次の入力に関する予測値が、現入力と等しい値に設定され、即ち、以下のとおりとされる。

【０１８１】

prediction = original

【０１８２】

デコーダ２０において、第１逆合算演算、即ち方法３が、有利には以下のとおり実施される。全てのデータ値に関し、入力値、即ち「ｏｒｉｇｉｎａｌ」（「元」）値に対応する入力値、即ち「ｒｅｓｕｌｔ」（「結果」）がソフトウェアループにおいて以下のとおり計算される。

【０１８３】

result = original - prediction

if result < lowValue then result = result + wrapValue

【０１８４】

最終的に、次の入力に関する予測値が、現結果と等しい値に設定され、即ち、以下のとおりとされる。

【０１８５】

prediction = original

【０１８６】

エンコーダ１０において、第２直接合算演算、即ち方法４が、有利には以下のとおり実施される。全てのデータ値に関し、入力値、即ち「ｏｒｉｇｉｎａｌ」（「元」）値に対応する入力値、即ち「ｒｅｓｕｌｔ」（「結果」）がソフトウェアループにおいて以下のとおり計算される。

【０１８７】

result = original + prediction

if result > highValue then result = result - wrapValue

【０１８８】

最終的に、次の入力に関する予測値が、現結果と等しい値に設定され、即ち、以下のとおりとされる。

【０１８９】

prediction = original

【０１９０】

デコーダ２０において、第２逆合算演算、即ち方法４が、有利には以下のとおり実施される。全てのデータ値に関し、入力値、即ち「ｏｒｉｇｉｎａｌ」（「元」）値に対応する出力値、即ち「ｒｅｓｕｌｔ」（「結果」）がソフトウェアループにおいて以下のとおり計算される。

【０１９１】

result = original - prediction

if result < lowValue then result = result + wrapValue

【０１９２】

最終的に、次の入力に関する予測値が、現結果と等しい値に設定され、即ち、以下のとおりとされる。

【０１９３】

prediction = original

【０１９４】

このような合算及び差分の演算、４つの方法すべてが、即ちＯＤｅｌｔａバージョンのエンコーダ１０及びデコーダ２０を実施する際に、１ビットデータ、即ちビット単位で適用可能でもある。１ビットデータの状況においては、次の値は、エンコーダ１０及びデコーダ２０の両方によって既知であり、つまり、ＭＩＮ＝０、ＭＡＸ＝１である。さらに、有利には、ｌｏｗＶａｌｕｅ＝ＭＩＮ＝０かつｈｉｇｈＶａｌｕｅ＝ＭＡＸ＝１と仮定される。さらに、このような場合、「ｎｕｍｂｅｒ」（「数」）は、従って、［ｈｉｇｈＶ

ａｌｕｅ－ｌｏｗＶａｌｕｅ＝１－０＝１］であり、ｗｒａｐＶａｌｕｅについては、「ｎｕｍｂｅｒ」（「数」）＋１＝１＋１＝２が選択されるという利点を持つ。その理由は、ｌｏｗＶａｌｕｅ＝ＭＩＮ＝０から開始する正の値のみを有し得ると考慮される１ビットデータのみ存在するからである。１ビットデータに関し、方法１及び方法３は、相互に類似の符号化結果を算出する。同様に、方法２及び方法４は、相互に類似の符号化結果を算出する。このような知識を有することは、様々なデフォルトが仮定され得るので、データＤ２に送信される必要のある情報を有利に簡素化する。即ち、差分演算、つまり方法１又は方法２の実行回数の数、及び選択された予測（入力値（方法１）又は結果値（方法２））についての情報を送信する必要があるだけで、その結果、デコーダ２０は、データＤ２、Ｄ３又はＤ４を復号して復号データＤ５を生成する際に、正しい逆差分演算を必要回数、実行し得る。

【０１９５】

同様の出力を作成する方法１又は方法３を用いることによって作成された第１の例は、やはり同様の出力を作成する方法２又は方法４を用いることによっても処理され得る。以下に示す結果は、それら方法をデータ式１に適用した場合に取得され得るものである。

【０１９６】

０１１００１０００１１１１００１１１１１１１１１１１１１０１０１０１０１０１

【０１９７】

この時、処理データは、２４個の「１」と１３個の「０」とを有し、即ちそのエントロピーは第１の例のものと同じであるが、「１」及び「０」のカウントは位置を変える。これは常に生じる訳ではなく、その代わり、しばしば、これら異なる方法の間でエントロピーが同様に変化する。例えば、データの最初の４つの要素の後、方法１及び／又は方法３は、３個の「１」と１個の「０」を生成するのに対し、元データと、方法２及び／又は方法４を用いて処理されたデータは、２個の「１」と２個の「０」とを有する。従って、このような場合の方法１及び／又は方法３は、方法２及び／又は方法４より小さいエントロピーを生成し、また、元のものよりも小さいエントロピーを生成する。

【０１９８】

マルチビットによる実施においては、データＤ１が－６４～＋６３の範囲の値を含む場合、ＭＩＮ＝－６４かつＭＡＸ＝６３である。ｌｏｗＶａｌｕｅ＝ＭＩＮかつｈｉｇｈＶａｌｕｅ＝ＭＡＸと仮定することによって、「ｎｕｍｂｅｒ」（「数」）＝１２７であり、ｗｒａｐＶａｌｕｅには、有利に１２８が選択される。しかしながら、データＤ１がランダムに変化する場合、「ｐｒｅｄｉｃｔｉｏｎ」（「予測」）値は、有利には、値［（ｗｒａｐＶａｌｕｅ＋１）÷２＋ｌｏｗＶａｌｕｅ＝６４＋－６４＝０］に設定される。

【０１９９】

第１値が例えば－１である場合、直接ＯＤｅｌｔａ方法１及び／又は方法２を用いた第１符号化値は－１－０＝－１となり、これに対応して、直接ＯＤｅｌｔａ方法３及び／又は方法４を用いた第１符号化値は－１＋０＝－１となるであろう。次の値は、次いで、どのようにデータが進行するかに応じて変化し、例えば第２値が５となる場合、直接ＯＤｅｌｔａ方法１は５－ －１＝６を生成し、直接ＯＤｅｌｔａ方法２は５－ －１＝６を生成し、直接ＯＤｅｌｔａ方法３は５＋ －１＝４を生成し、直接ＯＤｅｌｔａ方法４は５＋－１＝４を生成するだろう。デコーダ２０は、この場合、逆ＯＤｅｌｔａ方法１及び／又は方法２を用いた場合に第１値として－１＋０＝－１を生成することが可能であり、逆ＯＤｅｌｔａ方法３及び／又は方法４を用いて、－１－０＝－１を生成することが可能であるだろう。これに対応して、逆ＯＤｅｌｔａ方法１を用いた第２値は６＋ －１＝５となり、逆ＯＤｅｌｔａ方法２を用いた第２値は６＋ －１＝５となり、逆ＯＤｅｌｔａ方法３を用いた第２値は４－ －１＝５となり逆ＯＤｅｌｔａ方法４を用いた第２値は４－ －１＝５となるであろう。

【０２００】

この解決手段は、数の範囲が実際に－２０～＋２７の値のみを含む場合には、次いで最

適化され得る。本例の場合、例えば、ｌｏｗＶａｌｕｅ＝－２０及びｈｉｇｈＶａｌｕｅ＝２７を伝達することが可能である。両方が伝達される場合は、ｎｕｍｂｅｒ＝４７と計算可能であり、ｗｒａｐＶａｌｕｅには、次いで、有利には４８が選択される。すると、ｐｒｅｄｉｃｔｉｏｎについて、値４８÷２＋ －２０＝４を計算可能である。次いで、以前の例により、ＯＤｅｌｔａ方法１又は方法２が用いられる場合、例えば、値－１に関して－１－４＝－５が生成され、ＯＤｅｌｔａ方法３又は方法４が用いられる場合には、－１＋４＝３が生成される。同様に、第２値は、ＯＤｅｌｔａ方法については、（５－－１）＝６、（５－ －５）＝１０、（５＋ －１）＝４、及び（５＋３）＝８となるであろう。デコーダ２０は、再び正常に機能し、方法１及び／又は方法２に関して第１値を－５＋４＝－１と生成し、方法３及び／又は方法４に関して３－４＝－１を生成する。対応して、各種方法に関する第２値は、（６＋ －１）＝５、（１０＋ －５）＝５、（４－－１）＝５、及び（８－３）＝５として復号されるだろう。

【０２０１】

上記のこれら例における値は、上記範囲、即ち－６４～＋６３又は－２０～＋２７の内にあることが解り、従って、これらの例による値の内で補正項を実行する必要はないが、任意の負又は正の変化が十分に大きいものである場合には、データ値に対する補正が、与えられる式２１～２４によって為され、上記範囲内に結果値を維持する必要がある。ここで補正項はラップアラウンド値を指していることに留意されたい。

【０２０２】

ｌｏｗＶａｌｕｅが既知である場合、エントロピー符号化データＤ３と共にエンコーダ１０からデコーダ２０に送信されなければならない符号化テーブルを簡素化するために、符号化値は、有利には、０で開始し、かつ値「ｎｕｍｂｅｒ」（「数」）で終了するように構成される。この演算は、ポストオフセットと呼ばれ、このポストオフセット値は、エントロピー符号化の後、かつデータＤ４に対する逆ＯＤｅｌｔａ演算の前に、符号化データ値から削除されなければならない。

【０２０３】

当初述べたとおり、ｐｒｅ－ｏｆｆｓｅｔ（プリオフセット）機能を用いてオフセットを実施することも可能であり、この場合、元入力データ（Ｄ１）は、ＯＤｅｌｔａ方法の実際の実行の前に既に０から「ｎｕｍｂｅｒ」（「数」）までの値を含み得る正の要素に変換される。また、この状況においては、「プリオフセット」及びＯＤｅｌｔａ法が繰り返し同じ情報を伝達しないように、又は何らかの他の方法の結果、既知であることを無視するように、この演算が要する情報伝達が実行されるのが有利である。適切なＤ５出力データを作成するためには、このプリオフセットによる効果は、逆ＯＤｅｌｔａ演算の後に復号データから削除されなければならない。

【０２０４】

上記に記載した発明の実施形態に対する変更は、添付の特許請求の範囲によって定義される発明の要旨から逸脱することのない限り可能である。本発明を説明し、また特許請求する際に用いられる「含む」又は「備える」（”ｉｎｃｌｕｄｉｎｇ”、”ｃｏｍｐｒｉｓｉｎｇ”）、「組み込む」（”ｉｎｃｏｒｐｏｒａｔｉｎｇ”）、「から成る」（”ｃｏｎｓｉｓｔｉｎｇ　ｏｆ”）、「有する」（”ｈａｖｅ”）、「である」（”ｉｓ”）等の表現は、非排他的に解釈されるべきことが意図されており、即ち明示されていない事項、成分又は要素が存在し得る。単数での表記は、複数にも関連するものと理解される。添付の特許請求の範囲における括弧内の数値は、請求項の理解を補助するためのものであり、それら請求項によって記載される主題を限定するものとして解釈してはならない。

【０２０５】

本願出願当初の特許願に添付された特許請求の範囲に記載の実施形態を、以下に載せる。

［１］数値列を含む入力データ（Ｄ１）を符号化することにより、対応符号化出力データ（Ｄ２又はＤ３）を生成するエンコーダ（１０）であって、

前記エンコーダ（１０）は、差分符号化及び／又は合算符号化の方式を前記入力データ

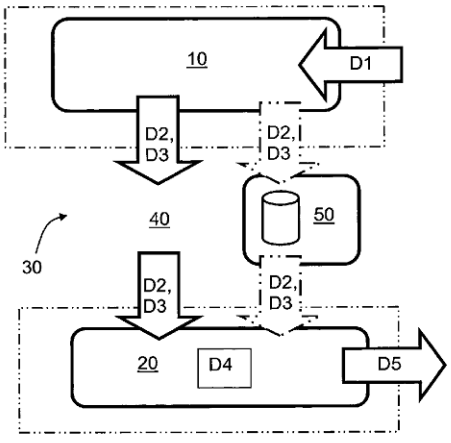
（Ｄ１）に適用することにより１つ又は複数の対応符号化列を生成するデータ処理装置を備え、
前記１つ又は複数の対応符号化列は、符号化出力データ（Ｄ２又はＤ３）を生成するために、最大値におけるラップアラウンド及び／又は最小値におけるラップアラウンドを受ける、
エンコーダ（１０）。
［２］前記データ処理装置は、前記入力データ（Ｄ１）及び／又は前記１つ又は複数の対応符号化列を分析することにより、符号化出力データ（Ｄ２又はＤ３）の生成に用いるための前記１つ又は複数の対応符号化列に適用する１つ又は複数のオフセット値、最小値又は最大値を計算するように動作可能であることを特徴とする、［１］に記載のエンコーダ（１０）
［３］前記１つ又は複数のオフセット値は、値「０」を有することを特徴とする、［２］に記載のエンコーダ（１０）。
［４］１つ又は複数の１ビット値を含む数値を処理するように動作可能であり、かつ、ビット単位方式で前記入力データ（Ｄ１）を符号化するように動作可能であることを特徴とする、［１］に記載のエンコーダ（１０）。
［５］前記１つ又は複数の対応符号化列は、前記入力データ（Ｄ１）の連続値における変化を表すことを特徴とする、［１］に記載のエンコーダ（１０）。
［６］別々に符号化される複数のデータセクションに前記入力データ（Ｄ１）を分割するように動作可能であることを特徴とする、［１］に記載のエンコーダ（１０）。
［７］前記データセクションに対して選択的に符号化を適用するように動作可能であり、該符号化の適用は、これによってデータ圧縮が符号化出力データ（Ｄ２又はＤ３）において達成可能である場合にのみ実行されるものである、［６］に記載のエンコーダ（１０）。
［８］出力符号化データ（Ｄ２又はＤ３）を作成するのに利用される一連の予測値としてデフォルト第１予測値を利用するように動作可能である、［１］に記載のエンコーダ（１０）。
［９］前記第１予測値が、０，最高値÷２，最高値÷２＋最低値，最高値－最低値のうちの少なくともいずれかである、［８］に記載のエンコーダ（１０）。
［１０］更なる符号化を適用することにより、符号化入力データ（Ｄ２）を生成し、該更なる符号化が、ランレングス符号化（ＲＬＥ）、可変長符号化（ＶＬＣ）、ハフマン符号化、算術符号化、レンジ符号化のうちの少なくとも１つを含むことを特徴とする、［１］から［９］の何れか１項に記載のエンコーダ（１０）。
［１１］相互に類似のビットのランレングスに応じて前記入力データ（Ｄ１）を複数のセクションに分割するように動作可能であり、前記ランレングスは、ランレングス符号化（ＲＬＥ）、ハフマン符号化、可変長符号化（ＶＬＥ）、レンジ符号化及び／又は算術符号化を用いる符号化に有効なものであることを特徴とする、［６］に記載のエンコーダ（１０）。
［１２］前記データ処理装置は、コンピュータハードウェアを用いて実装されるものであり、該コンピュータハードウェアは、機械可読データ記憶媒体に記録された１つ又は複数のソフトウェアを実行するように動作可能であることを特徴とする、［１］から［１１］の何れか１項に記載のエンコーダ（１０）。
［１３］数値列を含む入力データ（Ｄ１）を符号化するエンコーダ（１０）を用いることにより、対応符号化入力データ（Ｄ２又はＤ３）を生成する方法であって、該方法は、
（ａ）差分符号化及び／又は合算符号化の方式を前記入力データ（Ｄ１）に適用することにより、１つ又は複数の対応符号化列を生成することと；
（ｂ）符号化出力データ（Ｄ２又はＤ３）を生成するために、前記１つ又は複数の対応符号化列に対して最大値におけるラップアラウンド及び／又は最小値におけるラップアラウンドを行うことと；
を含む、方法。

［１４］前記入力データ（Ｄ１）及び／又は前記１つ又は複数の対応符号化列を分析することにより、符号化出力データ（Ｄ２又はＤ３）の生成に用いるための前記１つ又は複数の対応符号化列に適用する１つ又は複数のオフセット値、最小値、又は最大値を計算する前記データ処理装置を用いることを特徴とする、［１３］に記載の方法。
［１５］前記１つ又は複数のオフセット値に関し値「０」を用いることを含むことを特徴とする、［１４］に記載の方法。
［１６］１つ又は複数の１ビット値を含む数値を処理することを含み、
前記エンコーダ（１０）が、ビット単位方式で前記入力データ（Ｄ１）を符号化するように動作可能である、
［１４］に記載の方法。
［１７］前記１つ又は複数の対応符号化列は、前記入力データ（Ｄ１）の連続値における変化を表すことを特徴とする、［１３］に記載の方法。
［１８］データ処理装置を使って、別々に符号化される複数のデータセクションに前記入力データ（Ｄ１）を分割することを含むことを特徴とする、［１３］に記載の方法。
［１９］前記データセクションに対して選択的に符号化を適用することを含み、該符号化を適用することは、これによってデータ圧縮が符号化出力データ（Ｄ２又はＤ３）において達成可能である場合にのみ実行されるものであることを特徴とする、［１８］に記載の方法。
［２０］出力符号化データ（Ｄ２又はＤ３）を作成するのに利用される一連の予測値としてデフォルト第１予測値を利用することを含む、［１３］に記載の方法。
［２１］前記第１予測値が、０，最高値÷２，最高値÷２＋最低値，最高値－最低値のうちの少なくともいずれかである、［２０］に記載の方法。
［２２］更なる符号化を適用することにより、符号化出力データ（Ｄ２又はＤ３）を生成することを含み、該更なる符号化が、ランレングス符号化（ＲＬＥ）、可変長符号化（ＶＬＣ）、ハフマン符号化、算術符号化、レンジ符号化のうちの少なくとも１つを含むことを特徴とする、［１３］から［２１］の何れか１項に記載の方法。
［２３］コンピュータハードウェアを用いて前記処理装置を実施することを含み、該コンピュータハードウェアは、機械可読データ記憶媒体に記録された１つ又は複数のソフトウェアを実行するように動作可能であることを特徴とする、［１３］から［２２］の何れか１項に記載の方法。
［２４］符号化データ（Ｄ２又はＤ３）を復号することにより対応復号出力データ（Ｄ５）を生成するデコーダ（２０）であって、
前記符号化データ（Ｄ２又はＤ３）の１つ又は複数の部分を処理するデータ処理装置を備え、
前記データ処理装置は、前記１つ又は複数の部分の１つ又は複数の対応符号化列に対して差分復号及び／又は合算復号の方式を適用するように動作可能であり、
前記１つ又は複数の符号化列が、前記復号出力データ（Ｄ５）を生成するために、最大値におけるラップアラウンド及び／又は最小値におけるラップアラウンドを受ける、
デコーダ（２０）。
［２５］１つ又は複数の１ビット値を含む前記符号化データ（Ｄ２又はＤ３）を復号するように動作可能であり、かつ、ビット単位方式で前記符号化データ（Ｄ２又はＤ３）を復号するように動作可能であることを特徴とする、［２４］に記載のデコーダ（２０）。
［２６］前記復号出力データ（Ｄ５）の生成に用いるための前記１つ又は複数の符号化列に適用する１つ又は複数のオフセット値、最小値、又は最大値を完成するように動作可能であることを特徴とする、［２４］に記載のデコーダ（２０）
［２７］前記１つ又は複数のオフセット値が値「０」であることを特徴とする、［２６］に記載のデコーダ（２０）。
［２８］前記１つ又は複数の対応符号化列は、符号化データ（Ｄ２又はＤ３）に符号化される連続値における変化を表すことを特徴とする、［２４］に記載のデコーダ（２０）。
［２９］前記データ処理装置は、該装置を介して処理されるデータに対して、ランレング

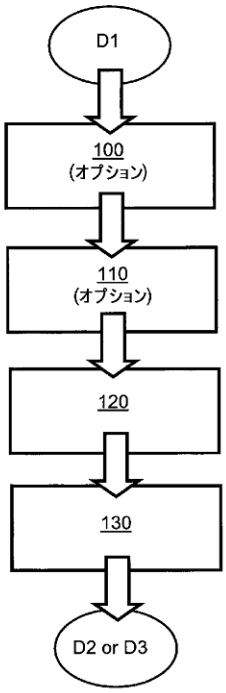
ス符号化（ＲＬＥ）、可変長符号化（ＶＬＣ）、ハフマン符号化、算術符号化、レンジ符号化の少なくとも１つの逆を適用するように動作可能であることを特徴とする、［２４］に記載のデコーダ（２０）。
［３０］前記データ処理装置は、該装置を介して復号される一連のデータにおいてデフォルト第１予測値を仮定するように動作可能である、［２４］に記載のデコーダ（２０）。
［３１］前記デフォルト値は、値「０」を有することを特徴とする、［３０］に記載のデコーダ（２０）。
［３２］前記データ処理装置は、コンピュータハードウェアを用いて実装されるものであり、該コンピュータハードウェアは、非一時的機械可読データ記憶媒体に記録された１つ又は複数のソフトウェアを実行するように動作可能であることを特徴とする、［２４］から［３１］の何れか１項に記載のデコーダ（２０）。
［３３］符号化データ（Ｄ２又はＤ３）を復号するデコーダ（２０）を用いることにより対応前記復号出力データ（Ｄ５）を生成する方法であって、
（ａ）データ処理装置を用いて、前記符号化データデータ（Ｄ２又はＤ３）の１つ又は複数の部分を処理することと；
（ｂ）前記データ処理装置は、前記１つ又は複数の部分の１つ又は複数の対応符号化列に対して差分復号及び／又は合算復号の方式を適用するように動作可能であり、前記１つ又は複数の符号化列が、前記復号出力データ（Ｄ５）を生成するために、最大値におけるラップアラウンド及び／又は最小値におけるラップアラウンドを受けることと；
を特徴とする方法。
［３４］１つ又は複数の１ビット値を含む前記符号化データ（Ｄ２又はＤ３）を復号するように前記デコーダ（２０）を動作させることを含み、
ここで前記デコーダ（２０）は、ビット単位方式で前記符号化データ（Ｄ２又はＤ３）を復号するように動作可能である、
［３３］に記載の方法。
［３５］１つ又は複数のオフセット値、最小値、及び最大値を完成するように前記データ処理装置を動作させることを含み、該１つ又は複数のオフセット値、最小値、及び最大値は、前記復号出力データ（Ｄ５）の生成に用いるための前記１つ又は複数の符号化列に適用するものであることを特徴とする、［３３］に記載の方法。
［３６］前記１つ又は複数のオフセット値が値「０」であることを特徴とする、［３５］に記載の方法。
［３７］前記１つ又は複数の対応符号化列は、前記符号化データ（Ｄ２又はＤ３）に符号化される連続値における変化を表すことを特徴とする、［３３］に記載の方法。
［３８］前記データ処理装置を、該装置を介して処理されるデータに対して、ランレングス符号化（ＲＬＥ）、可変長符号化（ＶＬＣ）、ハフマン符号化、算術符号化、レンジ符号化の少なくとも１つの逆を適用するように動作させることを含む、［３３］に記載の方法。
［３９］前記データ処理装置を、該装置を介して復号される一連のデータにおいて第１予測値のデフォルト値を仮定するように動作させることを含む、［３３］に記載の方法。
［４０］前記デフォルト値は、値「０」を有することを特徴とする、［３９］に記載の方法。
［４１］コンピュータハードウェアを用いて前記処理装置を実装することを含み、該コンピュータハードウェアは、機械可読データ記憶媒体に記録された１つ又は複数のソフトウェアを実行するように動作可能であることを特徴とする、［３３］から［４０］の何れか１項に記載の方法。
［４２］前記入力データ（Ｄ１）を符号化することにより対応符号化データ（Ｄ２）を生成する、［１］に記載のエンコーダ（１０）を少なくとも１つ、並びに前記符号化データ（Ｄ２又はＤ３）を復号することにより対応復号データ（Ｄ５）を生成する、［２４］に記載のデコーダ（２０）を少なくとも１つ備えるコーデック装置（３０）。
［４３］装置の処理手段に実行されることにより、前記装置に、［１３］～［２３］及び

［３３］～［４１］のいずれか１項に記載の方法を遂行させるように構成されるプログラム命令を備える、コンピュータプログラム。

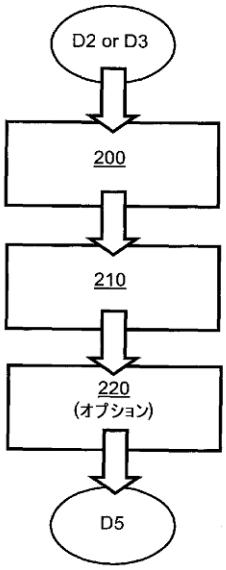
【図１】



【図２】



【図３】



フロントページの続き

審査官　北村　智彦

(56)参考文献　特開２００８－０６７３６１（ＪＰ，Ａ）
特開平０８－３０７２７９（ＪＰ，Ａ）
特開２００３－２４９８５６（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｈ０３Ｍ　　３／００－１１／００
ＩＥＥＥ　Ｘｐｌｏｒｅ
ＣｉＮｉｉ