(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2005-527911
(P2005-527911A)
(43)公表日　平成17年9月15日(2005.9.15)

(51)Int.Cl.[7]	F I		テーマコード（参考）
G06F 17/16	G06F 17/16	G	5B013
G06F 9/32	G06F 9/32	330A	5B033
G06F 9/38	G06F 9/38	330E	5B056
	G06F 9/38	330J	

審査請求 未請求　予備審査請求 有　（全 42 頁）

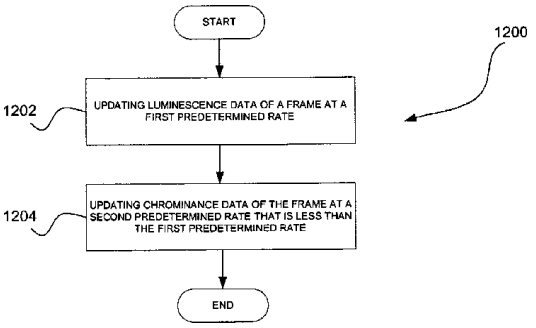
(21)出願番号　特願2004-508038 (P2004-508038)
(86)(22)出願日　平成15年5月28日 (2003.5.28)
(85)翻訳文提出日　平成17年1月31日 (2005.1.31)
(86)国際出願番号　PCT/US2003/016908
(87)国際公開番号　WO2003/100655
(87)国際公開日　平成15年12月4日 (2003.12.4)
(31)優先権主張番号　60/385,253
(32)優先日　平成14年5月28日 (2002.5.28)
(33)優先権主張国　米国 (US)
(31)優先権主張番号　60/385,250
(32)優先日　平成14年5月28日 (2002.5.28)
(33)優先権主張国　米国 (US)
(31)優先権主張番号　60/385,251
(32)優先日　平成14年5月28日 (2002.5.28)
(33)優先権主張国　米国 (US)

(71)出願人　504389500
ドロップレット　テクノロジー　インコーポレイテッド
アメリカ合衆国　カリフォルニア州　９５０１４　カペルティーノ　スティーヴンス　クリーク　ブールヴァード　２０３００　スイート　２５０
(74)代理人　100072051
弁理士　杉村　興作
(74)代理人　100100125
弁理士　高見　和明
(74)代理人　100101096
弁理士　徳永　博
(74)代理人　100086645
弁理士　岩佐　義幸

最終頁に続く

(54)【発明の名称】パイル処理・並列プロセッサ用のシステム及び方法

(57)【要約】
システム、方法及びコンピュータプログラムプロダクトは、例外処理（図１２）用に提供される。他のコーダとデコーダとの少なくとも一方のシステム及び方法は、可変モジュールを含めて提供される。さらに他のシステム及び方法は、圧縮データ用に提供され、それらによって、フレームのルミネッセンスデータを、予め設定された第一のレートでアップデートする（１２０２）とともに、フレームのクロミナンスデータを、予め設定された第一のレートとなる予め設定された第二のレートでアップデートする。



【特許請求の範囲】
【請求項１】
例外を処理する方法であって、
演算処理をループで行い、
前記演算処理をループで行う間に、前記例外を識別し、
前記演算処理をループで行う間に、前記例外を格納し、
前記例外を前記ループから分けて処理することを特徴とする方法。
【請求項２】
前記演算処理が、無効数字を含むことを特徴とする、請求項１記載の方法。
【請求項３】
前記計算に関する演算が、複数個のゼロを計数することを含むことを特徴とする、請求項２記載の方法。
【請求項４】
前記計算に関する演算が、クリッピングとサチュレーティングの少なくとも一つを含むことを特徴とする、請求項１記載の方法。
【請求項５】
前記例外が、有効数字を含むことを特徴とする、請求項１記載の方法。
【請求項６】
前記例外が、ゼロでないデータを含むことを特徴とする、請求項５記載の方法。
【請求項７】
前記演算処理を、変換モジュールを利用して少なくとも部分的に行うことを特徴とする、請求項１記載の方法。
【請求項８】
前記演算処理を、量子化モジュールを利用して少なくとも部分的に行うことを特徴とする、請求項１記載の方法。
【請求項９】
前記演算処理を、エントロピコードモジュールを利用して少なくとも部分的に行うことを特徴とする、請求項１記載の方法。
【請求項１０】
前記格納が、例外を処理するために要求される格納データを含むことを特徴とする、請求項１記載の方法。
【請求項１１】
前記処理を、データを圧縮するために実行することを特徴とする、請求項１記載の方法。
【請求項１２】
前記データを、逆相関変換を用いて圧縮することを特徴とする、請求項１１記載の方法。
【請求項１３】
前記データを、ウェーブレット変換を用いて圧縮することを特徴とする、請求項１１記載の方法。
【請求項１４】
前記データを、離散的コサイン変換を用いて圧縮することを特徴とする、請求項１１記載の方法。
【請求項１５】
例外を処理するコンピュータプログラムプロダクトであって、
演算処理をループで行うコンピュータコードと、
前記演算処理をループで行う間に、前記例外を識別するコンピュータコードと、
前記演算処理をループで行う間に、前記例外を格納するコンピュータコードと、
前記例外を前記ループから分けて処理するコンピュータコードとを具えることを特徴とするコンピュータプログラムプロダクト。

【請求項１６】
例外を処理するシステムであって、
変換モジュール、量子化モジュール及びエントロピコードモジュールからなる群から選択した少なくとも一つのデータ圧縮モジュールを具え、
前記少なくとも一つのデータ圧縮モジュールが、演算処理をループで行い、前記演算処理をループで行う間に、前記例外を識別し、前記演算処理をループで行う間に、前記例外を格納し、前記例外を前記ループから分けて処理するように適合されたことを特徴とするシステム。
【請求項１７】
可変モジュールを具えることを特徴とするコーダ。
【請求項１８】
前記モジュールが、圧縮アルゴリズムと関連した確率分布曲線の急勾配を反映することを特徴とする、請求項１７記載のコーダ。
【請求項１９】
前記モジュールが、確率分布の負の指数を有することを特徴とする、請求項１８記載のコーダ。
【請求項２０】
前記確率分布がコーデックに適合することを特徴とする、請求項１８記載のコーダ。
【請求項２１】
前記コーデックが、所定のパフォーマンスレベルを与えられた、最小限の計算上の複雑性を利用するために設計されることを特徴とする、請求項１８記載のコーダ。
【請求項２２】
前記モジュールが、データの前回の一式のコンテキストに依存することを特徴とする、請求項１７記載のコーダ。
【請求項２３】
前記モジュールが、実行長の関数に従う増加を回避することを特徴とする、請求項１７記載のコーダ。
【請求項２４】
前記実行長が、シーケンスで複数の同一ビットを有することを特徴とする、請求項２３記載のコーダ。
【請求項２５】
前記コーダが、エントロピコードを有することを特徴とする、請求項１７記載のコーダ。
【請求項２６】
可変モジュールを具えることを特徴とするデコーダ。
【請求項２７】
圧縮アルゴリズムと関連した確率分布曲線の急勾配を反映することを特徴とする、請求項２６記載のデコーダ。
【請求項２８】
前記モジュールが、確率分布の負の指数を有することを特徴とする、請求項２７記載のデコーダ。
【請求項２９】
前記確率分布がコーデックに適合することを特徴とする、請求項２７記載のデコーダ。
【請求項３０】
前記コーデックが、所定のパフォーマンスレベルを与えられた、最小限の計算上の複雑性を利用するために設計されることを特徴とする、請求項２７記載のデコーダ。
【請求項３１】
前記モジュールが、データの前回の一式のコンテキストに依存することを特徴とする、請求項２６記載のデコーダ。
【請求項３２】

前記モジュールが、実行長の関数に従う増加を回避することを特徴とする、請求項２６記載のデコーダ。
【請求項３３】
前記実行長が、シーケンスで複数の同一ビットを有することを特徴とする、請求項３２記載のデコーダ。
【請求項３４】
前記デコーダが、エントロピコードを有することを特徴とする、請求項２６記載のコーダ。
【請求項３５】
圧縮アルゴリズムと関連した確率分布曲線の急勾配を反映し、かつ、実行長とともに増加しない可変モジュールを具えるコーデックを用いる方法。
【請求項３６】
ビデオデータを圧縮する方法であって、
予め設定された第一のレートでフレームのルミネセンスデータをアップデートし、
前記予め設定された第一のレート未満である予め設定された第二のレートでフレームのクロミナンスデータをアップデートすることを特徴とする方法。
【請求項３７】
前記クロミナンスデータの一以上の周波数帯域を除外することを特徴とする、請求項３６記載の方法。
【請求項３８】
前記一以上の周波数帯域を、フィルタを用いて除外することを特徴とする、請求項３７記載の方法。
【請求項３９】
前記フィルタがウェーブレットフィルタを有することを特徴とする、請求項３８記載の方法。
【請求項４０】
さらに、ビデオデータの圧縮解除の際に前記クロミナンスデータの除外された部分を補完することを特徴とする、請求項３７記載の方法。
【請求項４１】
ビデオデータを圧縮するコンピュータプログラムプロダクトであって、
予め設定された第一のレートでフレームのルミネセンスデータをアップデートするコンピュータコードと、
前記予め設定された第一のレート未満である予め設定された第二のレートでフレームのクロミナンスデータをアップデートするコンピュータコードとを具えることを特徴とするコンピュータプログラムプロダクト。
【請求項４２】
前記クロミナンスデータの一以上の周波数帯域を除外することを特徴とする、請求項４１記載のコンピュータプログラムプロダクト。
【請求項４３】
前記一以上の周波数帯域を、フィルタを用いて除外することを特徴とする、請求項４２記載のコンピュータプログラムプロダクト。
【請求項４４】
前記フィルタがウェーブレットフィルタを有することを特徴とする、請求項４３記載のコンピュータプログラムプロダクト。
【請求項４５】
さらに、ビデオデータの圧縮解除の際に前記クロミナンスデータの除外された部分を補完することを特徴とする、請求項４２記載のコンピュータプログラム。
【請求項４６】
ビデオデータを圧縮する方法であって、
ビデオデータを圧縮し、

前記圧縮されたデータで停止情報を挿入し、
前記停止情報を、前記ビデオデータがその再生中に停止された場合に用いることを特徴とする方法。
【請求項４７】
前記停止情報を、再生されたビデオデータの質を向上させるために用いることを特徴とする、請求項４６記載の方法。
【請求項４８】
前記停止情報が、高解像度フレームを有することを特徴とする、請求項４７記載の方法。
【請求項４９】
前記停止情報が、高解像度フレームを構成するために用いることができるデータを有することを特徴とする、請求項４７記載の方法。

【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、データ処理に関する。
【背景技術】
【０００２】
並行処理
並列プロセッサは、要求されるアルゴリズムが、狭いデータ幅、シリアルデータ依存性又は頻出する制御ステートメント(例えば「if」、「for」、「while」ステートメント)を有する場合、高い処理能力を目的としてプログラミングをすることが難しい。プロセッサにおけるそのような問題を克服するために用いられる３タイプの並列処理がある。
【０００３】
第1のタイプの並列処理は、複数の関数ユニットによってサポートされ、処理をそれぞれの関数ユニットにおいて同時進行可能とする。スーパースケーラープロセッサ・アーキテクチャ及び超長命令語(VLIW：very long instruction word)プロセッサー・アーキテクチャーは、同じサイクル上でいくつかの関数ユニットのそれぞれに命令を出すようにする。一般的に、ラテンシーすなわち終了に要する時間は、あるタイプの機能ユニットと別のタイプの機能ユニットとで異なる。最も簡単な機能（例えば、ビットワイズAND）は、浮動の加算機能が３又はそれよりも多いサイクルを処理する間に、大抵１サイクルで完了する。
【０００４】
第２のタイプの並列処理は、個々の機能的な複数ユニットをパイプライン型に設けることによってサポートされる。例えば、浮動ADDは、終了するまでに３サイクルを要するとともに、１サイクルごとに必要とする三つのシーケンシャルサブ機能に取り込み可能である。サブ機能間にパイプライン型に複数レジスタを設置することによって、第二の浮動ADDは、先の浮動ADDを第二サブ機能に取り込むのと同じサイクル上で第一サブ機能に取り込み可能である。この手段によって、たとえ任意の個々の浮動ADDが終了するために３サイクルを要するとしても、浮動ADDを各サイクルで開始し及び終了させることができる。
【０００５】
利用可能な第３のタイプの並列処理は、同じ計算の互いに相違するインスタンスに応じて１ワードの異なるフィールド・パーティションを当てるものである。例えば、３２ビットプロセッサの３２ビットワードは、８ビットの四つのフィールド・パーティションに分割することができる。もし、データ項目が８ビット内に収めるのに十分小さければ、同じ単一の命令で四つの値全てを処理することができる。
【０００６】
さらに、フィールド・パーティションの数と機能ユニット開始数との積に等しい複数のデータ項目を単一サイクルごとに処理することができる。

【０００７】
ループ・アンローリング
多重化とパイプライン化との少なくとも一つを行う機能ユニットをプログラミングする従来の一般的な取り組みがあり、それは、同じ計算の複数のインスタンスを見つけるとともに、各インスタンスから対応する動作を同時に実行するものである。複数のインスタンスは、ループ・アンローリングの周知の技術によって、又は同一計算の別のソースによって作成することができる。
【０００８】
ループ・アンローリングは一般的に適切な技術であるが、具体例は、利点を習得することにおいて役立つ。例えば、以下のプログラムAを考察する。
【０００９】
プログラムＡ

```
for i = 0:1:255, { S(i) };
```

【００１０】
ここに、本文S(i)は、iに依存する複数処理の一部の配列 {S1(i); S2(i); S3(i); S4(i); S5(i);} であり、計算S(i)は、計算S(j)に完全に独立であり、j≠iである。処理S1(i); S2(i); S3(i); S4(i); S5(i);は互いに独立であるように仮定しない。それと反対に、ある処理から次の処理への依存によって並び変えを禁止していると仮定する。
【００１１】
さらに、これら同じ依存性のものは、前の処理を完了するまで、次の処理を始めないように要求することを仮定する。もし、パイプライン型処理の各々が完了するために（たとえ、パイプライン型実行ユニットが各サイクルで新しい結果を生んだとしても）２サイクルを要するならば、五つの処理のシーケンスは、完了するために１０サイクルを要する。加えて、プログラミングツールが分岐遅延（branch delay）でS4(i); S5(i);をオーバーラップできなければ、ループ分岐は、典型的に、ループごとに付加的な３サイクルを要する。したがって、プログラムAは、分岐遅延をオーバーラップする場合、完了するために６４０（２５６／４＊１０）サイクルを必要とし、分岐遅延をオーバーラップしない場合、完了するために８３２（２５６／４＊１３）サイクルを必要とする。
【００１２】
下記プログラムＢはプログラムＡと等価である。
【００１３】
プログラムＢ

```
for n = 0:4:255, { S(n); S(n+1); S(n+2); S(n+3);};
```

【００１４】
ループが、４回「アンローリング」される。これは、四つの因子によって、不経済な制御フロー変化の数を減少する。より重要なことに、四つのS(i)のそれぞれの連続処理を記録するための機会が得られる。したがって、プログラムA及びプログラムBはプログラムＣと等価である。
【００１５】
プログラムＣ

```
for n = 0:4:255, { S1(n); S2(n); S3(n); S4(n); S5(n);
S1(n+1); S2(n+1); S3(n+1); S4(n+1); S5(n+1);
S1(n+2); S2(n+2); S3(n+2); S4(n+2); S5(n+2);
S1(n+3); S2(n+3); S3(n+3); S4(n+3); S5(n+3);
};
```

【００１６】
上記で独立・従属についての仮定一式を有する場合、等価プログラムDを作ることができる。

【００１７】
プログラムＤ

```
for n = 0:4:255, { S1(n); S1(n+1); S1(n+2); S1(n+3);
S2(n); S2(n+1); S2(n+2); S2(n+3);
S3(n); S3(n+1); S3(n+2); S3(n+3);
S4(n); S4(n+1); S4(n+2); S4(n+3);
S5(n); S5(n+1); S5(n+2); S5(n+3);
};
```

【００１８】
第１サイクルでS1(n); S1(n+1); が現れ、第２サイクルでS1(n+2); S1(n+3); が現れる。第３サイクルの開始でS1(n); S1(n+1) が完了（２サイクルが経過）することで、S2(n); S2(n+1); が現れる。したがって、次の二つの処理がそれぞれ次のサイクルで現れることで、本文全体が同じ１０サイクルで実行される。プログラムDは、プログラムAの４分の１の時間で処理する。したがって、ループ展開の周知の利益が示される。

【００１９】
大抵の並列プロセッサは、命令それ自身と分岐が実際に起こるポイントとの間の遅延でいくつものサイクルを要求する、従来の分岐命令をもつ必要がある。この遅延期間中、他の命令を実行することができる。分岐は、分岐条件が十分に早く分かっているとともに、コンパイラ又は他のプログラミングツールが、命令の実行を遅延の間サポートする限り、分岐はせいぜい一命令出現機会（issue opportunity）くらいのコストしかかからない。この技術は、分岐条件（i=255）がループの先頭で分かっているとき、プログラムＡに適用できる。

【００２０】
しかし、アンローリング実行は逆効果を生じ得る。第１に、一旦、複数出現機会の全てが（プログラムDにおけるように）利用されると、付加的アンローリングで更に加速することはない。第２に、アンロールされたループターンの各々は、一般的に特定ターンを理由として状態を保つために、付加的な複数レジスタを必要とする。必要とされるレジスタの数は、アンロールされるターンの数に正比例する。必要とされるレジスタの全体数が利用可能な数を超える場合、レジスタの一部が、キャッシュ（cache）に送出され、その後、次のループターン上で、もとに戻される。送出及びリロードをサポートするために発せられることを要求される複数命令は、プログラム作成時間（program time）を長くする。したがって、そのようなループをアンロールするための最適回数がある。

【００２１】
例外処理を含むアンローリングループ
ここでプログラムA’を考察する。

【００２２】

プログラムA’

```
for i = 0:1:255, { S(i); if C(i) then T(I(i)) };
```

【００２３】
ここで、C(i)が、S1(i)だけに従属する、何らかの例外的な（約６４中１）真の除外状態であり、そしてT(I(i))は、約１０２４処理の冗長な例外処理である。I(i)は、例外処理に要求されるS(i)によって算出された情報である。例えば、プログラムAのループターンごとに、ループの主本文内の４処理を超える量である、平均して１６処理を加算するT(I(i))を規定する。そのようにまれにしか起こらない非常に長い例外処理は、アンローリングの利益を失うことなしに処理する方法が明らかにできない点で共通のプログラミングの問題である。
【００２４】
保護された命令
そのような問題に対処するアプローチの一つは、保護された命令、複数プロセッサ上で利用可能な設備の使用を通じたものである。保護された命令は、命令が常に予測された機能ユニットを占有する意味で、加算オペランド（additional operand）としてブール値を指定するが、保護が偽（false）ならば、結果保持が抑制される。
【００２５】
「if-then-else」を実行中、保護は、「if」条件であるように選ばれる。「then」節の命令が「if」条件によって保護されるとともに、「else」節の命令が、「if」条件の否定によって保護される。いずれの場合も、両方の節が実行される。保護が「真（true）」である命令だけが、「then」節の結果によってアップデートされる。さらに、保護が「偽（false）」である命令だけが、「else」節の結果によってアップデートされる。全インスタンスが両方の節の命令を実行し、パイプラインが、制御フローの各条件付変化によって要求されるペナルティを遅延するのではなく、このペナルティに持ちこたえる。
【００２６】
保護されたアプローチは、プログラムA’におけるように、主に「true」及び「else」節が大きい場合、大きなペナルティを受ける。その場合に、全インスタンスは、たとえほんの少数がそれによって影響を受けたとしても、大きい「else」節ペナルティを受ける。もし、条件Ｃによって保護されるための処理Ｓを持っているならば、「guard(C,S);」としてプログラムすることができる。
【００２７】
第一アンローリング
プログラムＡ’を、次のようなプログラムＤ’にアンロールすることができる。
【００２８】

```
for n = 0:4:255, { S1(n); S1(n+1); S1(n+2); S1(n+3);
S2(n); S2(n+1); S2(n+2); S2(n+3);
S3(n); S3(n+1); S3(n+2); S3(n+3);
S4(n); S4(n+1); S4(n+2); S4(n+3);
S5(n); S5(n+1); S5(n+2); S5(n+3);
if C(n) then T(I(n));
if C(n+1) then T(I(n+1));
if C(n+2) then T(I(n+2));
if C(n+3) then T(I(n+3));
};
```

【００２９】
上記複数パラメータ例を与えると、T(I(n))をループターンの77%で実行しなくても良くなり、一つのT(I(n))を、ループターンの21%、かつ、ループターンのわずか２％について一以上のT(I(n))で、実行すればよい。明らかに、T(I(n))、T(I(n+1))、T(I(n+2))及びT(I(n+3))の処理をインターリービングによって得られることはほとんどない。
【００３０】
したがって、例外処理のための向上技術が必要である。
【００３１】
コーデック
エンコーダは、別のものに入力一連シンボルを位置づけるものであり、コード化され、デコーダと呼ばれる他のプロセッサがコード化された一連の記号から入力される一連の記号を再構成できるような方法で一連の記号を再構成する。エンコーダ及びデコーダペアはコーデックと呼ばれる。
【００３２】
簡単明瞭にいうと、有限である一連の記号はよく文字列と呼ばれるため、一つが入力文字列及びコード化文字列に参照される。入力文字列の各記号は、関連付けられ、有限のアルファベット記号Iから取り出される。同様に、コード化された文字列の各記号は、関連付けられ有限のアルファベットのコード記号Cから取り出される。
【００３３】
各アルファベットは、<end>記号と呼ばれる識別記号を含む。（a distinguished symbol）文字列は関連付けられた<end>記号で終了するとともに、<end>記号は一文字列の終端だけに現れればよい。<end>記号の目的は、コーデック処理を規則正しく停止させることである。
【００３４】
入力又はコード文字列の終了を決定する任意の方法を、実の又は仮想の<end>記号の作用を合成するために用いることができる。例えば、複数のアプリケーションで、少なくとも入力及びコード化のいずれか一方をされた文字列の長さが知られ、その情報は、文字通りの<end>文字列用の置換に使用することができる。
【００３５】
エンコーダマッピングはΦで表示することができ、そのため、ｕが入力文字列であるとともにｖが対応コード化文字列である場合、一つはｖ＝Φ(u)と書くことができる。同様に、デコーダマッピングは、Ψで表示することができ、u＝Ψ(Φ(u))の場合には、ｕ＝Ψ(v)と書くことができる。
【００３６】
ｖを再構成するためにΦ(Ψ(v))を必要とすることはない。関連アルファベットＩ及び

Ｃがそれぞれ、<end>記号に加えてちょうど二つの記号を含む場合、コーデック(Φ，Ψ)はバイナリコーデックと呼ばれる。ａ、ｂ及び<end>が、バイナリアルファベット内の三つの記号である場合、有用な関数（useful function）～は、～a = b, ～b = a, ～b = a, ～<end> = <end> であるように定義される。

【００３７】

コーデックは、これまで説明したように、入力文字列数（及びコード文字列の数）が無限大になるように実現されない。コーデック上で追加の構造及び限定を設定しなければ、限られたマシンで容易に実行できず、実現されていない。

【００３８】

コーデックの重要な一部（subset）は、周知の有限状態変換器によって実際に実行可能とされることである。有限状態変換器（FST: Finite State Transducer）は、連続するようにコード文字列の記号を書いて、その開始記号からその終了記号<end>までの文字列を順次に処理するオートマトン（automaton）である。情報は、入力文字列の記号から順次得られるとともに最終的にコード文字列で表現される。入力文字列を得ることとそれをコード文字列に表現することとの間の遅れを調整するために、FSTは、連続するように、状態を維持するとともにアップデートする。状態は、状態スペースと呼ばれる可能な状態の有限セットから選択される。状態スペースは、<start>及び<finish>と呼ばれる二つの区別された状態を含む。FSTは、<start>状態でその処理を開始し、<finish>状態でその処理を終了する。<finish>状態は、<end>記号が入力文字列から読み取られるまで到達することはなく、かつ、<end>記号はコード文字列に付加される。

【００３９】

状態スペースが有限であるから、FSTとしてあらゆるエンコーダを表すことはできない。実用性の理由で、現在の記述は、エンコーダ及びデコーダの両方がFSTSとして記述され、かつ、実行することができるコーデックに集中している（focus）。FSTとしてエンコーダΦを実行することができる場合、それはアップデート関数φによって定めることができる。入力文字列から第1の入力記号ａが、カレント状態slと結合し、そして、次の状態s2を作成する。第1の記号は、入力文字列の始めから条件付きで取り除かれる。作成されたコード記号bは、コード文字列に条件付きで付加される。

【００４０】

カレント状態が<finish>で、かつ、FSTが順次（sequencing）終了する場合、関数φは不確定である。要約のため、$(s_2, b) = (\phi_s(s_1, a), \phi_b(s_1, a)) = \phi(s_1, a)$とする。ここに、$\phi_s(s_1, a)$は、定義上は$\phi(s_1, a)$の第1成分であり、そして、$\phi_b(s_1, a)$は、定義上は$\phi_b(s_1, a)$の第2成分である。

【００４１】

エントロピコーディングを含む多くのアプリケーションのために、FSTにマルコフ確率構造（Markovian probability structure）を備えることが有益である。状態s_1及び入力記号aを与えると、確率$Prob(a|s_1)$ができ、ここではFSTが状態s_1で与えられ、ａが次の読み取り記号となる。アプリケーションに応じて、この確率を、規定することができ、ヒストリカル・データから静的に概算する又はＦＳＴの最近の動作から動的に見積もることができる。この後者のケースでは、確率の見積もりに基づく情報を、状態スペースでコード化することができる。

【００４２】

これから、FSTが状態s_1で与えられ、次に状態s_2となる場合には、確率$Prob(s_2|s_1)$を計算することができる。これは、ケース分析として：

$$Prob(s_2|s_1) = (\phi_s(s_1, a) = s_2)\ Prob(a|s_1) + (\phi_s(s_1, \sim a) = s_2)\ Prob(\sim a|s_1)$$

によって計算される。

【００４３】

マルコフ状態遷移確率のこの集合(set)は、$M_{ij} = Prob(s_j|s_i)$である確率マトリックスMに組み込むことができる。漸近的状態確率P(s)は、最大固有値1に対応するMの右固有ベクトル(right eigenvector)の要素として計算することができる。

【００４４】
視覚的な見地
ビデオの「コーデック」(コンプレッサー／デコンプレッサ(decompressor))は、イメージ品質と、プロセッサ必要条件(すなわち、コスト／電力消費)と、圧縮比(すなわち、結果的に得られるデータ速度)と間で均衡を保つことにより、データ通信ストリームに要求されるデータ速度を小さくするために用いられる。現在のところ利用可能な圧縮アプローチは、トレード・オフの異なる範囲を提示し、複数のコーデック・プロファイルを生じさせ、各プロファイルが特定アプリケーションの要求を満たすように最適化される。
【００４５】
ロッシィ（Lossy）デジタルビデオ圧縮システムは、著しく小さいデジタル表示を作り出すためにデジタル化されたビデオシーケンス上で作動する。可視的な結果は、オリジナルのビデオに酷似しているように見えるが、一般的に、完全に一致しないことがある。これらのシステムに関し、プロセスで失われた情報が、ビューアーによって容易に見られないかすぐに気づかれないビデオの外観に一致することが重要である。
【００４６】
典型的なデジタルビデオ圧縮システムは、変換段階、量子化段階及びエントロピコード化段階を具える、一連の段階で作動する。MPEG及び他のＤＣＴベースコーデック・アルゴリズム(DCT-based codec algorithms)のようないくつかの圧縮システムは、動き補償検索などのような他の段階を加え、２Ｄ及び３Ｄのウェーブレットが、DCTベースコーデック・アルゴリズムの現在の代替である。JPEG委員会が依然として画像規格であるそのＪＰＥＧ２０００のためのウェーブレットアルゴリズムを採用することを促すことで、ウェーブレットはそれらの満足のいくイメージ品質及びフレキシブルな圧縮比により高く評価されている。
【００４７】
ビデオコンプレッサーで、変換段階としてウェーブレット変換を用いる場合、そのようなアルゴリズムは、高帯域成分及び低帯域成分すなわちデータを分けたフィルタ・ペアのシーケンスとして作動する。標準のウェーブレット変換は2次元的な方法で、単一イメージの空間的な広がりで作動する。2次元は、垂直的に機能する(work vertically)フィルタと水平的に機能する(work horizontally)フィルタとを組み合わせることによって処理される(handle)。典型的に、これらは、厳密な交替を必要としないが、順次、Ｈ－Ｖ－Ｈ－Ｖを交替する。時間方向(temporal direction)でも同様に：調子を合わせた連続的な画像からのサンプルで作動することで、ウェーブレットフィルタ（wavelet filter）を適用することが技術的に知られている。さらに、ウェーブレット（wavelet）変換は、ビデオ信号の、明度又はビデオ信号の輝度(luma)及び色差又はクロミナンス(色度)要素に、別々に適用することができる。
【００４８】
時間方向で、ウェーブレット型変換と一緒に、ＤＣＴ又は空間2-D (spatial 2-D)用の他の非ウェーブレット空間変換を用いることができる。これは混合3-D変換(mixed 3-D transform)は、3-Dウェーブレット変換と同じ目的を果たす。また、3D-ＤＣＴ変換のために時間方向で短いＤＣＴを用いることができる。
【００４９】
３-Ｄウェーブレット変換の時間的部分は、典型的には、はるかに短くなる点で、空間的部分と異なる。空間的変換のための典型的サイズは、水平に７２０の画素であり、垂直に４８０の画素；空間変換用の典型的な大きさが、２、４、８又は１５フレームである。これらの時間的長さは更に小さくなる。なぜならば、処理中に遅れが長くなる結果（それは好ましくないが）として処理する多くのフレームが、それらが処理される間、フレームを格納することが必要となり、これは不経済である。
【００５０】
その品質を判断するために、１画像(picture)若しくは１ビデオシーケンスを見る場合又は２画像若しくは２ビデオシーケンスを視覚的に比較する場合、欠陥又は違いを検知す

ることは、他のものより難しい。このことは、見るもののいくつかの外観(aspects)に対して、他のものよりも高い感度を有するという、人間の視覚系統の結果である。例えば、それらがハイコントラストである場合に限り、人は非常に細かな細部を見ることができる一方、非常に微妙な中規模の細部を見ることができる。これらの違いは圧縮のために重要である。圧縮プロセスは、違い及びエラーをできるだけ目立たなくするように設計される。したがって、圧縮プロセスは、細かい細部でより多くのエラーを許容しながら、明度コントラストの中間サイズで優れた忠実度を実現できる。

【００５１】

したがって、知覚される質を著しく犠牲にすることなく、圧縮アルゴリズムを向上する様々な心理物理学的な機会（psychophysics opportunities）を活用することが要求され続けている。

【００５２】

上記圧縮システムは、パーソナルビデオレコーダー、デジタルビデオレコーダー、ケーブル・セットトップ・ボックス、その他同種類のものでよく使用される。これらアプリケーションなどの共通の特徴は、ユーザが、ビデオを停止したり、静止画像として長時間（extended time）単一フレームを表示させたままにしたりする可能性がある。

【００５３】

入力が画像より高解像度の単一画像を取得するため、ビデオシーケンス又は画像の他のシーケンスを処理することが技術的に知られている。しかしながら、この処理は、場面中で動いているもの、カメラ動作、明暗変動（lighting shifts）及び他の変化を識別し、又は調和をとらなければならないとともに、それぞれの変化を、個々にかつ組み合わせて補正しなければならないので、計算に際して非常に不経済である。それにもかかわわらず、現代の機器は、簡単な停止機能に関するそのような計算上の浪費を現在サポートしていない。

【００５４】

したがって、圧縮解除されたビデオのフレームを単に繰り返すことによって作り出されるよりも視覚的に品質の十分に高い静止画像を示すための様々な心理物理学的な機会を活用することが要求され続けている。

【発明の開示】

【課題を解決するための手段】

【００５５】

例外処理

例外処理のシステム、方法及びコンピュータプログラムプロダクトを提供する。最初に、演算処理をループで行う。さらに、前記演算処理を行う間に、前記例外を識別するとともに格納する。その後、そのような例外を前記ループから分けて処理する。

【００５６】

一例では、前記演算処理が、無効数字（non-significant values）を含んでいてもよい。例えば、前記演算処理が、複数個のゼロを計数することを含んでいてもよい。
なおかつさらに、前記演算処理が、クリッピングとサチュレーティングの少なくとも一つを含んでいてもよい。

【００５７】

他の例では、前記例外が、有効数字を含んでいてもよい。例えば、前記例外が、非ゼロデータを含んでいてもよい。

【００５８】

選択肢として、前記演算処理を、例えば、データ圧縮システムの、変換モジュールと量子化モジュールとエントロピコードモジュールとの、少なくとも一つを利用して部分的に行ってもよい。このようにして、前記処理を、データを圧縮するために実行してもよい。任意に、前記データを、ウェーブレット変換と、離散的コサイン変換（discrete cosine transforms）と、他のタイプの逆相関変換（de-correlating transform）との少なくとも一つを用いて圧縮してもよい。

【００５９】
コーデック
可変モジュール具える、コーダとデコーダとの少なくとも一方のシステム及び方法を提供する。
【００６０】
一例では、前記モジュールが、圧縮アルゴリズムと関連した確率分布曲線の急勾配を反映してもよい。例えば、前記モジュールが、確率分布の負の指数を有してもよい。選択肢として、前記確率分布がコーデックに適合してもよい。
【００６１】
他の例では、前記モジュールが、データの前回の一式のコンテキストに依存してもよい。さらに、前記モジュールが、実行長（すなわち、シーケンスで複数の同一ビット）の関数に従う増加を回避してもよい。
【００６２】
さらに、他の例では、前記コーデックが、所定のパフォーマンスレベルを与えられた、最小限の計算上の複雑性を利用するために設計されてもよい。
【００６３】
視覚的な見地
圧縮データのシステム及び方法を提供する。使用に際し、予め設定された第一のレートでフレームのルミネセンスデータをアップデートするとともに、前記予め設定された第一のレート未満である予め設定された第二のレートでフレームのクロミナンスデータをアップデートする。
【００６４】
したがって、圧縮量が増加する。これを達成するために、一例では、前記クロミナンスデータの一以上の周波数帯域を除外してもよい。さらに、前記一以上の周波数帯域を、フィルタを用いて除外してもよい。そのようなフィルタが、ウェーブレットフィルタを有してもよい。したがって、ビデオデータの圧縮解除の際に、前記クロミナンスデータの除外された部分が補完される。
【００６５】
圧縮データの他のシステム及び方法を提供する。そのようなシステム及び方法は、ビデオデータを圧縮すること及び圧縮したデータを有する停止情報を挿入することを具えてなる。このようにして、停止情報は、前記ビデオデータがその再生中に停止された場合に用いられる。
【００６６】
一例では、前記停止情報が、停止作動中に、再生されたビデオデータの質を向上させるために用いられてもよい。さらに、前記停止情報が、高解像度フレームを有していてもよい。なおかつさらに、前記停止情報が、高解像度フレームを構成するために用いることができるデータを有してもよい。
【発明を実施するための最良の形態】
【００６７】
図1は、一実施の形態による、データ圧縮／圧縮解除の構成１００を示す。この構成１００は、ともに「コーデック」を構成するコーダ部１０１及びデコーダ部１０３を有する。コーダ部１０１は、ファイル１０８中に格納するデータを圧縮する変換モジュール１０２、量子化器１０４及びエントロピエンコーダ１０６を有する。そのようなファイル１０８の圧縮解除を実行するために、デコーダ部１０３は、使用(すなわち、ビデオデータなどの場合には見ること)するデータを圧縮解除するエントロピ・デコーダ１１０、逆量子化器１１１及び逆変換モジュール１１４を有する。
【００６８】
使用に際し、変換モジュール１０２は、非相関関係の目的のために、多くの場合線形的に、複数の画素（すなわち、ビデオデータの場合）の可逆的変換を実行する。次に、量子化器１０４は、変換値の量子化を行い(effect)、その後、エントロピエンコーダ１０６が

、量子化された変換係数のエントロピコーディングを担う。デコーダ部１０３の様々な要素は、原則的にそのような処理を逆にする。

【００６９】

図２は、一実施例に従う例外処理の方法２００を示す。一実施例では、本方法２００を、図１の構成１００のコンテキストで実行する。しかしながら、方法２００を、任意に要求されるコンテキストで実行することができる。

【００７０】

最初に、オペレーション２０２で、演算処理をループ内で行う。ここで記載のコンテキストでは、演算処理が無効数字（non-significant value）を含む。例えば、演算処理が、多くの場合データ圧縮過程の間に実行される複数個のゼロを計数することを含む。さらに、演算処理が、データ圧縮のコンテキストで、クリッピングとサチュレーティングの少なくとも一つを含む。いずれの場合でも、演算処理は、他の値よりも有効でない任意の値の処理を含んでいてよい。

【００７１】

演算処理をループ内で行う間、例外を工程２０４～２０６で識別し及び格納する。選択的に、格納には、例外を処理するために要求される任意に関連するデータを格納することを含めることができる。本記載のコンテキストで、例外には、有効値を含めることができる。例えば、例外は、ゼロでないデータを有していてもよい。どんな場合でも、例外は、他の値よりも重要である任意の値の処理を有することができる。

【００７２】

このようにして、例外は、ループから分けて処理される。処理２０８参照。このために、例外処理は、ループのアンローリングをできるようにすることによってループの「パイル」処理を妨げないとともに、分岐をさせることによる後の向上パフォーマンスを妨げない。本実施例は、特に、冗長な例外クローズ（clauses）の並列実行をできるようにする。このことは、メモリに又はメモリからのデータを書き込み及び再読み込みすることによって達成させることができる。そのような技術と様々なオプションに関するより多くの情報及び「パイル」処理を、以下でより詳細に述べる。

【００７３】

オプションとして、各種の処理２０２～２０８は、データ圧縮システムの変換モジュール、量子化モジュール及びエントロピコードモジュールの少なくとも一つを利用して部分的に行えばよい。例えば、図１の構成１００の各種のモジュール参照。このように、処理２０２～２０８は、圧縮／圧縮解除データのために実行される。選択的に、データを、ウェーブレット変換、離散的コサイン変換（ＤＣＴ）及び任意の他の好適な逆相関変換（de-correlating）変換の少なくとも一つを利用して圧縮してもよい。

【００７４】

図３は、図２の方法２００の代表的な処理３００を示す。本図を、図２の方法２００のコンテキストで記載するが、代表的な処理３００は、任意の好適なコンテキストで実施することができる。

【００７５】

図のように、オペレーショナル計算３０４の第一スタック３０２は、ループ３０６で処理するために規定される。オペレーショナル計算３０４のその第一スタック３０２を通る処理の間、種々の例外３０８が、分離スタック３０８に格納されるとともに、別々に処理される。例えば、例外３０８は、分離ループ３１０のコンテキストで処理される。

【００７６】

任意の実施の形態

ここで、図２の処理のコンテキストで実施される「パイル」処理の様々な任意の特性に関するより多くの情報を示す。ここで記載のコンテキストでは、「パイル」は、メモリ（すなわち、ＲＡＭ）に記憶させることができる逐次記憶オブジェクト（a sequential memory object）である。パイルを、逐次書き込まれるとともに、開始から順次に続いて読み込まれるよう意図することができる。複数の方法は、パイルオブジェクトで規定される。

【００７７】
並列処理環境で実行されるパイル及びそれらの方法に対して、インライン（すなわち、分岐をサブルーチンに戻さない）コードの多少の命令で実現することができる。また、このインラインコードが分岐命令を含まないようにすることができる。そのような方法の実行を以下に記載する。パイルを特に有益にするように実現することができる。
【００７８】
表１は、一実施の形態によるパイル処理を実行するために行われる様々な処理を示す。
【００７９】
表１
１）　一つのパイルはCreate_Pile(P)方法によって作られる。これは、記憶域を割り当てするとともに内部状態変数を初期化する。
２）　パイルに書き込む第一の方法は、Conditional_Append(pile,condition,record)である。この方法は、状態が真(true)の場合に限りレコードをパイルに付加する。
３）　パイルが完全に書き込まれた場合に、Rewind_Pile(P)方法によって読込みに備える。これは、書かれた第一レコードで始まる読込みのための内部変数を整える。
４）　方法EOF(P)は、パイルのレコードの全てが読まれたかどうかを示す、ブール値（Boolean value）を作る。
５）　方法Pile_Read(P, record)はパイルPから次に続くレコードを読む。
６）　方法Destroy_Pile(P)は、その状態変数の全ての割り当てを取り消すことによって、パイルPを破壊する。
【００８０】
スプリット・オフ状態処理のためのパイル使用
一例として、パイルPを用いて、プログラムD’（背景部分参照）を以下のプログラムE’に変換することができる。
【００８１】
プログラムE’

```
Create_Pile (P);
for n = 0:4:255, { S1(n); S1(n+1); S1(n+2); S1(n+3);
S2(n); S2(n+1); S2(n+2); S2(n+3);
S3(n); S3(n+1); S3(n+2); S3(n+3);
S4(n); S4(n+1); S4(n+2); S4(n+3);
S5(n); S5(n+1); S5(n+2); S5(n+3);
Conditional_Append(P, C(n), I(n));
Conditional_Append(P, C(n+1), I(n+1));
Conditional_Append(P, C(n+2), I(n+2));
Conditional_Append(P, C(n+3), I(n+3));
};
Rewind(P);

while not EOF(P) {
Pile_Read(P, I);
T(I);
};
Destroy_Pile (P);
```

【００８２】
プログラムE’は、パイルP上の例外計算Ｔに要求される情報を保存することによって動作する。例外状態Ｃ(n)に対応するＩレコードは、Ｐにおけるレコードの数字（例えば、１６）が、オリジナルプログラムＡ（背景部分参照）でループの繰り返し数（例えば、２５６）よりも少なくなるように、書き込まれる。
【００８３】
その後、分離「while」ループが、例外計算Ｔの全てを実行するパイルＰを最後まで読む。Ｐは、Ｃ(n)が真（true）である場合のためだけにレコードＩを含むから、それらの場合だけが処理される。
【００８４】
第２ループは、第１ループより難解となり得る。なぜならば、第２ループの繰り返し数（この例では平均で１６であるが）が不定だからである。それゆえ、「for」ループではなく「while」ループが用いられ、そしてそのループは、エンドオブファイル（EOF）法が、全レコードがパイルから読まれたことを示した場合に終了する。
【００８５】
上記説明及び以下の記載のように、Conditional_Append法を、インラインで分岐無しに実施することができる。これは、第１ループが、非生産問題がほとんどないという好機を有し、有効な方法でアンロールされたままであることを意味する。
【００８６】
第２ループのアンローリング
上記プログラムE’における第２ループは、アンロールされないが、依然として効率的ではない。しかしながら、四つのパイル手段P1, P2, P3, P4によってプログラムE’を以下のプログラムＦ’に変形することができる。その結果、プログラムＦ’が、付随する効率向上を有してアンロールされた両ループを持つ。
【００８７】

プログラムＦ’

```
Create_Pile (P1); Create_Pile (P2); Create_Pile (P3);
Create_Pile (P4);
for n = 0:4:255, { S1(n); S1(n+1); S1(n+2); S1(n+3);
S2(n); S2(n+1); S2(n+2); S2(n+3);
S3(n); S3(n+1); S3(n+2); S3(n+3);
S4(n); S4(n+1); S4(n+2); S4(n+3);
S5(n); S5(n+1); S5(n+2); S5(n+3);
Conditional_Append(P1, C(n), I(n));
Conditional_Append(P2, C(n+1), I(n+1));
Conditional_Append(P3, C(n+2), I(n+2));
Conditional_Append(P4, C(n+3), I(n+3));
};
Rewind(P1); Rewind (P2); Rewind (P3); Rewind (P4);

while not all EOF(Pi) {
Pile_Read(P1, I1);Pile_Read(P2, I2);
Pile_Read(P3, I3);Pile_Read(P4, I4);
guard(not EOF(P1), S);T(I1);
guard(not EOF(P2), S);T(I2);
guard(not EOF(P3), S);T(I3);
guard(not EOF(P4), S);T(I4);
};
Destroy_Pile (P1); Destroy _Pile (P2); Destroy _Pile (P3);
Destroy _Pile (P4);
```

【００８８】

プログラムＦ’はアンロールされた第２ループを持つプログラムＥ’である。アンローリングは、プログラムＥ’の単一パイルを四つのパイルに分けることによって達成され、その各々を互いに独立して処理することができる。プログラムＦ’における第２ループの各繰り返しは、これら四つのパイルの各々から１レコードを処理する。各レコードは独立して処理されるから、各Ｔの演算は、他の三つのＴのオペレーションをインターリーブできる。

【００８９】

「while」ループの制御は、全パイルが処理されるまでループに変更を加える。さらに、一般的に、全パイルが同じループターンで完結させる必要がないから、「while」ループ本文内の複数のＴが保護される。二つのパイル内のレコードの数が互いに大きく異なる限り、効率が悪くなり得るが、確率（すなわち、大数の法則）は、パイルがレコードの同じ数字を含んでいるということである。

【００９０】

当然、このパイリング技術は、再帰的に適用される。Ｔ自身が、冗長な状態の複数のＴ節を含む場合、ある付加的なパイルで第２ループから複数のＴが分割するとともに、第３ループをアンロールする。多くの実用的なアプリケーションは、数個の重ね合わせられた

例外節を有している。

【００９１】

パイル処理の実行

パイルオブジェクトの実行及びその方法は、上記で定められた実行基準に合わせるために簡単なままである。例えば、実行方法は、Create_Pile及びDestroy_Pile以外に、インラインコードのほんの数命令である。さらに、実行は、分岐命令を含まない。

【００９２】

本質的には、一つのパイルは、メモリ（すなわち、ＲＡＭ）内の割り付された線形配列及びポインタを有し、インデックスのカレント値は、読み又は書きのための次のレコード位置である。配列の書き込まれる大きさszは、パイルを書き込んでいる間に値がインデックスの最大値となるようなポインタである。ＥＯＦ法は、インライン状態

$(sz \leq index)$

のとき実行させることができる。ポインタベースは、パイルに書くための第一位置を指し示す値を有する。それは、Create_Pile法によって設定する。

【００９３】

Conditional_Append法は、インデックスの値で始まるパイル配列のためにレコードをコピーする。それから、インデックスは、0かレコードサイズ（sz _ index）かのいずれかである算定量によって区切られる。パラメータ状態が、真としての１の値及び偽としての０の値を持つから、インデックスは、index = index + condition*sz_recordのとき、分岐無しで計算される。

【００９４】

当然、この計算の多くの変化が存在し、それらの多くは、変数の複数重なった所定の特別な値を含んでいない。また、保護として、guard(condition, index = index + sz_record)を用いて計算できる。

【００９５】

レコードは、状態に関係なくコピーされる。状態が偽の場合、このレコードは非常に近接するレコード(the very next record)によって上書きされる。状態が真の場合、非常に近接するレコードはカレントレコードに従い書き込まれる。この非常に近接するレコードはその後、そのレコード自身によって上書きされ又は上書きされない。その結果、レコードが読まれるとともに処理される場合、それが再計算される所定の（すなわち、冗長な）データを意味するとしても、できるだけパイルに書き込まれないことが一般的に最適である。

【００９６】

リワインド法（The Rewind method）は、sz = index; index = baseによって簡単に実行される。このオペレーションはEOF法用に書かれたデータ量を記録するとともに、その後、開始インデックスをリセットする。

【００９７】

Pile_Read法は、（length sz_recordの）パイルの次の部分をIにコピーするとともに、次のようなインデックス：index = index + sz_record に区切る。Destroy_Pile はパイルの格納の割り当てを取り消す。これら技術の全て（Create_Pile及びDestroy_Pile）が、少数のインライン命令で、かつ、分岐無しに実行される。

【００９８】

フィールド・パーティションでのプログラミング

大きいがまれである「else」節(the large but “ else ” clause)の場合、保護された処理に替わるものは、パイル処理である。各例が開始したとき、「else」節は、入力データをアドレス可能なメモリ（すなわち、キャッシュ又はＲＡＭ）内のパイルに移送(transfer)する。コンテキストの一例では、パイルは、入力データに付加されたファイルのように振る舞う。これは、ポインタによって与えられたアドレスでメモリに書き込まれることに

よって達成される。ファイル処理では、ネクストライト（the next write）がちょうど終了したものに付加されるため、その後、ポインタが、書かれたデータのサイズだけ増分される。パイル処理では、ポインタの増分を保護状態にすることができる。保護が真である場合、ネクストライトは、ちょうど終了したものに付加される。保護が偽である場合、ポインタは増分されず、ネクストライトはちょうど終了したものにオーバーレイする。保護が例外的に真である場合、パイルは短く、かつ、「else」オペレーションを持つパイルの順次処理は、インスタンスの総数でなく真である保護（すなわち、if条件が偽）の数に比例する時間を要する。トレード・オフ（The trade-off）は、「else」オペレーションにおける蓄積（savings）対パイルの読み書きの追加のオーバーヘッドである。

【００９９】

多くのプロセッサは、様々な算術的かつ論理的なオペレーションを１ワードのばらばらなフィールド・パーティションで独立かつ並列に実行することができる特別な命令を有する。カレントディスクリプション（記述）は、各フィールド・パーティションごとに「bit-at-a-time」処理の方法を含む。現行例として、四つの８ビット・フィールド・パーティションを持つ３２ビットワードを有する例を考察する。１フィールド・パーティションの８ビットは、ワード内で連続するように選択され、その結果、「adds」が実行されるとともに、「carry’s」が単一フィールド・パーティション内を伝播する。通常利用できる算術的フィールド・パーティション命令は、フィールド・パーティションの最上位ビット（ＭＳＢ：Most Significant Bit）から次の最上位フィールド・パーティションの最下位ビット（ＬＳＢ：Least Significant Bit）へのcarry-upを妨げる。

【０１００】

例えば、全て等しい長さB及びワード長さの除数（divisor）を仮定する。さらに、フィールド・パーティションは、アルゴリズムの独立のインスタンスに供される。各命令によって同時に単語のフィールドの全てをを処理する所定の技術及びコード・シーケンスを、以下に示す。これら技術及びコード・シーケンスは、制御の変化を避けるために表２の技術を用いる。

【０１０１】

表２

A) 論理的／算術的計算で制御の変化の置換。例えば、a<0の場合、c=bとし、それ以外では、b*(a<0)は c=d が、c=（a<0 ? b:c）によって置換される。これは、c=b*(a<0)+d*(1-(a<0))によって順々に置換される。

B) 可変の値の置き換えを条件付きで抑制するための論理値を使用。

a<0の場合、c=b

c=b*(a<0)+c*(I-(a<0))となる。

プロセッサは、しばしば、この技術を実行する保護された命令が標準装備になっている。

C) b*(a<0)がb&(a<0 ? 0xffff : 0x0000)となる条件を課すための論理命令を使用（フィールド例は、16ビットかつ定数が16進法である。）。

D) 論理値を記憶アドレス及び配列添え字（array subscript）に用いる。これは、順番に書かれた添え字インデックスの前へ進むのを条件付きで抑えるパイリング技術を具えている。例えば、

a<0の場合、{c[i]=b; i++}

c[i]=b; i+= (a<0)となる。

この場合、コードの２要素は、厳密には等価でない。配列ｃは、最後に特別な保護インデックスを必要とする。ユーザは、iの最終値(final value)を調べることによって、ｃの直前値（last value）を切り捨てるかどうか見分けがつく。

【０１０２】

アッド／シフト（Add／Shift）

算術的に区切られたプロセッサは、しばしば、各フィールドが独立して行動するADD命令を有する。これらプロセッサの一部は、別の種類のfield-by-field命令（例えば、右に

シフトする分割算術右シフトが、あるフィールドを他のフィールドにシフトしないとともに、フィールドのＭＳＢ、サインビットを、ちょうど空いたＭＳＢにコピーする。）を有する。

【０１０３】

比較及びフィールドマスク

これらプロセッサの中には、マルチプル・コンディション・ビットを生成することによって、field-by-field 比較命令を有するものがある。そうでなければ、分割減数命令(the partitioned subtract instruction) は、多くの場合、この機能のサービスに押し込まれる(service)。この場合、a<bは、真を示すマイナス記号及び偽を示すプラス記号で、a-bとして計算される。フィールドの他のビットは関連がない。その結果、十分長いシフトを持つ分割算術右シフトによって、表２のＣ）における例を用いると、真である全ての１又は偽である全てのゼロのフィールドマスクに変換させることができる。この結果、二つの命令でマルチフィールド比較(a multi-field comparison)になる。

【０１０４】

分割算術右シフトが有効でない場合、フィールドマスクを、現代の全てのプロセッサで使われる４つの命令を用いて、符号ビット(sign bit)から構成することができる。これらは、表３に示される。

【０１０５】

表３

１．u = u & 0x8000によって無関連ビットをゼロにセット。

２．フィールド v = u>>15 のＬＳＢにシフト（１６ビット用論理右シフト）。

３．フィールドマスク w = (u-v) |uを作成。

４．xがゼロの場合に限り記号ビットがゼロであるため、ポジティブフィールドｘで、分割ゼロテストを、x + 0x7fffによって実行できる。フィールドが符号化された場合、一例としてx |x + 0x7fffを用いることができる。記号ビットは、既に説明したように、フィールドマスクに変換させることができる。

【０１０６】

当然、全フィールドがゼロの条件は、フィールドの（分割されていない）ワード合計とゼロとを比較することによって、単一命令でテストさせることができる。

【０１０７】

表現

所定の定数を規定することが有益である。各フィールド・パーティションのＭＳＢポジションで「１」を除くゼロワードを、ＭＳＢという。各フィールド・パーティションのＬＳＢポジションで「１」を除くゼロワードを、ＬＳＢという。ビット・パーティションにおけるビット数はＢである。別の方法で指定しない場合は、全ワードは符号化されず（Uint）、全右シフトは、ゼロを左に充填する論理である。

【０１０８】

マルチ・ビット・フィールド・パーティションにおける単一情報ビットは、多くの異なる方法で表現される。マスク表現は、互いに等しくかつ情報ビットに等しい所定のフィールド・パーティションの全ビットを有する。当然、ビット情報は、１ワード内であるフィールド・パーティションから他のフィールド・パーティションに変化してもよい。

【０１０９】

他の有益な表現は、ＭＳＢ表現である。情報ビットがフィールド・パーティションに対応するＭＳＢポジションに格納されるとともに、フィールド・パーティションビットのリマインダがゼロである。同様に、ＬＳＢ表現は、ＬＳＢポジションにおける情報ビット及び他のゼロ全てを有する。

【０１１０】

他の有益な表現は、ＭＳＢ表現であり、この場合、ゼロ情報ビットが、フィールド・パーティションの各ビットのゼロによって表現され、それ以外の場合には、「１」情報ビットによって表現される。マスク、ＭＳＢ及びＬＳＢ表現の全てはＺＮＺ表現であるが、必

ずしも逆の場合は当てはまらない。

【０１１１】

変換

表示間の変換は、少数のワード長の命令に対して一つを要するが、これら命令は、フィールド・パーティションの全てを同時に処理する。

【０１１２】

MSB-> LSB

一例として、ＭＳＢ表現ｘを、ワード論理右シフト命令 y = (((Unit) x) >> B) によってＬＳＢ表現yに変換することができる。ＬＳＢ表現ｘを、ワード論理左シフト命令 y = (((Unit) x) << B) によってＭＳＢ表現yに変換する。

【０１１３】

Mask-> LSB

マスク表現mを、non-MSBビットをクリアすることによってＭＳＢ表現に変換することができる。大抵のプロセッサでは、１ワードの全フィールド・パーティションを、単独「andnot」命令

$$(m \wedge \sim MSB)$$

でマスクからＭＳＢに変換することができる。

【０１１４】

同様に、マスク表現を、単独「andnot」命令

$$(m \wedge \sim LSB)$$

によってＬＳＢ表現に変換することができる。

【０１１５】

MSB-> Mask

MSB表現xからマスク表現zへの変換は、ワード長さ命令を用いて、次の手順で処理される。表４参照。

【０１１６】

表４

１．MSB表現xをLSB表現yに変換。

２．vを与えるためのｘからｙのワード減算。これは、ゼロであるＭＳＢビットを除外するマスクである。

３．マスク結果ｚを与えるためのｖとｘとのワードＯＲ論理。総手順は、

$$z = (x - (x \gg B)) \vee x$$

である。

【０１１７】

ZNZ-> MSB

１ワードのフィールド・パーティションの全ては、次のように、ZNZ xからMSB yに変換させることができる。MSB位置にゼロビットを有するとともにそれ以外では“１”ビットを有するワードにＺＮＺを加算するワード加算命令を用いることができる。この加算の結果、MSB位置に適切なビットを持つが、その他のビット位置は、何も持たない。これは、non-MSBビットをクリアするための「andnot」命令を適用することによって、補われる（remedy）。

$$y = (x + \sim msb) \wedge \sim MSB$$

【０１１８】

その他

その他の表現は、上記のようにMSB表現から見い出す(reach)ことができる。

【０１１９】

ビット出力

あるアプリケーション（例えば、エントロピコーデック）では、所定のビットを一個ずつ(one - by - one)ビットストリングの終わりまで付加することによって、ビットストリングを形成することが望まれる。本明細書では、フィールド・パーティション並列方法においてこれを行う方法を示す。フィールド・パーティション及び関連ビットストリングは、互いに独立し、並列命令の各表現に独立する。

【０１２０】

処理は、表５に示される次の方法を扱う(work)ことができる。

【０１２１】

表５

１．入力ビット及び有効な条件の両方が、マスク表現で生じる。

２．情報ビットは、フィールド・パーティションが満たされるまで、条件付きで（すなわち、有効条件が真）付加される。

３．フィールド・パーティションが満たされた場合、それは、対応するフィールド・パーティション・ストリングスに付加される。通常、フィールド・パーティションの長さは、全て等しく、かつ、ワード長の約数である。

【０１２２】

まだ完全に満たされていない独立したフィールド・パーティションは、単独ワードで抑制され、アキュムレータと呼ばれる。関連ビットポインタ・ワードがあり、そのワードの各フィールド・パーティションが単独1ビット（すなわち、残りのゼロ）を含んでいる。その単独１ビットは、そのフィールド・パーティション用に次の付加ビットを受け取るためのアキュムレータにおけるビット位置に対応するビット位置にある。アキュムレータのフィールド・パーティションが完全に満たされた場合、フィールド・パーティションは、対応するフィールド・パーティション・ストリングに付加されるとともに、アキュムレータ・フィールド・パーティションはゼロにリセットされる。

【０１２３】

情報ビット出力

入力される情報ビットを（条件付きで）付加することができる。入力ビットマスク、バリッド・マスク及びビットポインタは、互いにワードワイズ「ANDed」であり、さらに、アキュムレータと「ORed」である。これは、大抵のプロセッサで１ワードにつき３命令実行を要する。

【０１２４】

ビットポインタのアップデート

ビットがビットストリングのＬＳＢ終了時に付加されると仮定すると、フィールド・パーティションのＬＳＢでアップデートされていないビットポインタは、フィールド・パーティションが充填されていることを示す。どんな場合でも、ビットポインタ・ワードは、ビットポインタ右の１位置のバリッド・フィールド・パーティションごとに回すことによってアップデートされる。これを行う方法は、下記表６のとおりである。

【０１２５】

表６

ａ） ビットポインタをＬＳＢビット及びｎｏｎ－ＬＳＢビットに分ける（２ワード「AND」命令）。

ｂ） ＬＳＢでないビットワードを右に一つ論理シフトするワードで表現（１ワード「SHIFT」命令）

ｃ） ＬＳＢでないビットワードをＭＳＢポジション方向左に論理シフトするワードで表現（１ワード「SHIFT」命令）

ｄ） ｂ）及びｃ）の結果を一緒に「OR」ワードで表現（１ワード「OR」命令）。

ｅ） ｄ）の結果及びオリジナルビットポインタのビットワイズを一緒にmux（マルチプレクサ）する。muxを制御するバリッドマスク使用（大抵のプロセッサで１「XOR」，２「AND」及び１「OR」ワード命令）。

【０１２６】

上記記載のように、フィールド・パーティションは、ビットポインタ ｐの対応フィールド・パーティションがＬＳＢパーティションで１を有する場合に充填される。アキュムレータ充填の任意のフィールド・パーティションが、ゼロでないビットポインタ ｐのＬＳＢビットのみのワードによって示される。

$$f=(p \wedge LSB);\quad full=(f \neq 0)$$

【０１２７】

fullの確率は、大抵、パイリングのアプリケーションがすぐに使える状態であるため、0.5より著しく小さい。パイルA1の長さは、ビット付加オペレーションの数より著しく小さい。パイリングは、処理ループ全体に含まれるもの以外に処理が他の制御フロー変化を含む必要がないように設計される。

【０１２８】

その後、パイルA2は、A2のアイテムを通したループによって処理される。インデックスIは、関連A2が付加されるビットストリング配列を選択するために用いられる。ビットのフィールド・パーティションBは、通常、２つの便利な指数であるように選ばれる（例えば、8又は16ビット）。8ビット又は16ビットの数値用の格納命令は、それら長さを使いやすくする。ベーシックループ以外の制御変化は、必ずしも上記処理を通すことを要求されない。

【０１２９】

ビット・フィールド・スキャニング

コーデックに要求される通常のオペレーションは、１ワードの１フィールドにおけるビットの順次読み出しである。フィールドｘから取り出されるビットは、bit_pointerによって、単一「１」ビットを除く複数個の０（例えば、０ｘ０２００）のフィールド値に設計される。「１」ビットは、引き出されるビットで揃えられ、その結果、ｘ＆bit_pointerは、ビットから読み出される値に従ってゼロとなり又はゼロでなくなる。これは、上記のようにフィールドマスクに変換させることができる。このシーケンスの各命令は、１ワードのフィールド全部を同時に処理する。

【０１３０】

順次スキャニングは、適切な指示でbit_pointerをシフトするとともに適切な終了条件まで繰り返すことによって達成される。フィールドが同一ビット位置で終了するとは限らないので、未終了のフィールドが出力をしない間、終了したフィールドが出力をしないように、上記手順を変更する。これは、フィールドが未終了の場合には全て「１」であり又はフィールドが終了した場合には全て「０」であるバリッド・フィールド・マスクを作ることによって達成される。実際のスキャニングは、全フィールドが終了するまで続けられ、全てゼロのワードであるバリッド（valid）によって示される。

【０１３１】

終了条件は、多くの場合、terminal_bit_pointerのフィールド内の「１」ビットによって示されるポジションに到達するbit_pointerのビットである。これは、bit_pointer&terminal_bit_pointernaino「１」ビットによって示される。これらフィールドは、上記バリッド・フィールド・マスクに変換させることができる。

【０１３２】

本明細書は、多くの直列的従属（sequential dependencies）及びスキャンされる各ビット位置の制御フロー変化を有するが、このループは、要求される実際の計算時間を最小限に抑えるようにアンロールさせることができる。ビット・フィールド・スキャニングの通常のアプリケーションでは、フィールド全てが、フィールドの全てに共通するループ終了条件に導いている同数のビットを有する。

【０１３３】

フィールド・パーティションの適合型サブ・フィールド

関連ビットストリングのワードｗの各フィールド・パーティションのビット・ポジションc:dを付加させたい場合には、定数ｃを、各フィールド・パーティションのビット・ポジションｃの「１」を除いたゼロワードにさせる。同様に、定数ｄを、各フィールド・パーティションのビット・ポジションｄの「１」を除いたゼロワードにさせる。さらに、次のオペレーションを実行する。表７参照。

【０１３４】

表７

A) bit_pointer qをcに初期化。q = c;

A1) CONDを全てtrueに初期化。

B) ワードワイズ bitand(ビット論理積) q with w

$u = q \wedge w$

uは、ZNZ表現になる。

C) uを、ZNZ表現からマスク表現vに変換。

D) ここでvは、上記したようにビットストリング出力になる。全てtrueのＣＯＮＤを使用。

E) cond = (q==d)処理が行われたか、そうでなければ、ワードワイズ論理の場合、ステップB)に戻ってqを右に一つ(q>>1)ループをシフト。

【０１３５】

(d-c)の平均値は、多くの場合、エントロピコードアプリケーションに対してかなり小さい。処理E)のテストは、処理E)及び分岐パイプライン遅れをカバーするのに利用できる処理B)～D)を遅らせる分岐を持つ処理B)と同じくらい早く開始させることができる。また、サブ・フィールドが一致するので、フィールド・パーティションの単一ワード用の命令内部で連続的従属をカバーするための数ワード処理をアンロールすることは、比較的容易である。

【０１３６】

フィールド・パーティションのサブ・フィールドの不一致

c及びdがフィールド・パーティションによって変化する場合には、c及びdは上記のままであるが、上記処理E)のテストは、ワードの全フィールド・パーティションについて同じであるのではなく、フィールド・パーティションによって変化する。この場合、全フィールド・パーティションが終了するまで、終了したフィールド・パーティションについてのスキャン・アウトを遊休させるのが望まれる。一例として、上記手順を表８における次のように変更することが望ましい。

【０１３７】

表８

１） ステップＤ）は、フィールド区分値が完全フィールド区分に対して偽であるとともに不完全フィールド区分に対して真である状態を必要とすることがある。これは、コードワードをCONDに対して”andnot”命令する処理を処理Ｅ）に追加することによって達成される。

$COND = (COND \wedge \sim cond)$

２） CONDが全て偽でない場合、ステップＥ）におけるｉｆ状態を、Ｂ）に戻るように変更する必要がある。

したがって、以下の処理を開始する。

Ａ） ビットポインタｑをｃに初期化。ｑ＝ｃ

Ａ１） CONDを全て真に初期化。

Ｂ） ｑ及びｗのワードワイズ(wordwise)bitand

u＝q∧w

ｕをＺＮＺ表示にする。

Ｃ）　ｕをＺＮＺ表示からマスク表示ｖに変換。

Ｄ）　ｖを、既に説明したようなビットストリング出力とすることができる。全て真のCONDの使用。

Ｅ１）　cond=(q==d);

COND＝(COND∧～cond);

Ｅ２）　COND==0の場合、処理を行う。それ以外の場合、ワードワイズ論理シフトｑright one(q<<1)を処理Ｂ）に戻す。

【０１３８】

２値から１値へ－ビットフィールドカウントダウン

エントロピ符号化における共通動作は、フィールドを２値から１値に変換する動作であり、これは、値がｎであるフィールドに対して、０が続くｎ個の１のストリングを生成する。大抵のアプリケーションにおいて、ｎの値は、１の平均を有する負の指数分布を有すると予測され、その結果、平均的には、出力において最後の０に付加された一つの“１”を有することを予測することができる。

【０１３９】

複数の０を導くフィールド区分並列法(filed-partition parallel method)は、次の通りである。既に説明したように、ｃを、ワードＷの各フィールドのＭＳＢ位置の“１”を除いて全て０にする。ｄを、各フィールドのＬＳＢ位置の“１”を除いて全て０にする。Diff＝ｃ－ｄとする。maskをdiffに初期化する。

【０１４０】

手順は、当該フィールドを（並列に）カウントダウンし、それと同時にＭＳＢ位置ｃを最初にキャリーアップする。ＭＳＢ位置が減算後に“１”となる場合、フィールドの以前の値は０でなく、“１”を出力すべきである。ＭＳＢ位置が減算後に“０”となる場合、フィールドの以前の値が０となり、０を出力すべきである。任意の場合において、ＭＳＢ位置は、ワードＸの対応するフィールド区分に対して出力すべきビットを有する。

【０１４１】

一旦、フィールドが０に到達し、最初の０が出力されると、０の更なる出力を抑制することができる。Ｘの互いに相違する区分が、互いに相違する値及びビット数とは異なる出力を有することがあるので、小さい値を有するフィールド区分からの出力を、全てのフィールド値が０に到達するまで抑制することができる。このような抑制は、既に説明したように、ビット出力手順に対するマスク入力によって実現される。一旦、フィールド区分に対する最初の０が出力されると、マスクの対応するフィールド区分が０に戻され、更なる出力を抑制する。

【０１４２】

diffが各フィールド区分に対して同一である通常の場合において、diffを０に変更する必要がない。それ以外の場合、diffをマスクとともにＡＮＤ処理することができる。表９参照。

【０１４３】

表９

White mask ≠0

X=X+dii

Y＝ZNZ_2_mask(c∧X)

。この場合、ZNZ_2_maskを、上記変換をマスクするZNZとする。

$X = X \wedge \sim c$

Yを、既に説明したようなマスクを伴って出力。

$mask = mask \wedge Y$

飛び越しに対する典型的な供給ラテンシー(pipeline latency)の場合、見積もられた確率に従って上記ループを展開することを理解することができる。

【０１４４】

区分された演算に対するループ展開の最適化

while c,{s}の形態のループを有する場合、ｉ回目の繰返しのc==真の確率をP_iとし、演算ｃ及び戻りのコストをC(c)とし、演算のコストをC(s)とする。ｓの他の実行は、演算の出力に影響を及ぼさないが、コストC(s)を被る。

【０１４５】

ループをｎ回展開することができ、その結果、演算は、s;s;s;…s;while c,{s}となり、この場合、while loopに先行するｓのｎ回の実行が存在する。全コストは、表１０に示したようになる。

表１０

$$nC(s)+\left(C(c)+P_n\left(C(s)+C(c)+P_{n+1}(\ldots)\right)\right)=$$
$$nC(s)+C(c)+\left(P_n+P_nP_{n+1}+\ldots\right)\left(C(c)+C(s)\right)\approx$$
$$\approx(n-1)\alpha+U_n=TC(n,\alpha)$$

$$where\quad U_n=\left(P_n+P_nP_{n+1}+\ldots\right)\quad and\quad \alpha=\frac{C(s)}{C(c)+C(s)}$$

【０１４６】

一例として、ワードごとにｋ個の独立したフィールドを有し、Ｐが個別のフィールドの各々に戻る確率であると仮定する。この場合、

$$P_n=1-\left(1-p^n\right)^k$$

となる。

【０１４７】

図４は、一実施の形態による、P_nを例証するグラフ４００である。図５は、一実施の形態による、対応U_nを例証するグラフ５００である。各図における曲線は、k=1に相当するブルーを有するｋの値に対応する。

【０１４８】

図６及び７は、それぞれ、α＝０．３及びα＝０．７に関するトータルコストTC(ｎ,α)を示すグラフ６００及び７００である。図８は、一実施の形態による、最小限トータルコスト

$$\min_n\left(TC(n,\alpha)\right)=\overline{TC}(\alpha)$$

（破線）及び初期ループアンロールの最適数

$$\bar{n}(\alpha)$$

を示す、グラフ８００である。

【０１４９】

実例

エントロピコーディング適用に際しては、出力ビットは、１である確率０．５及びゼロである確率０．５を有する。しかもそれらは独立である。これら前提で、一例では次の計算を行う。

【０１５０】

所定フィールド・パーティションが、ｎ以下の出力ビット（終了ゼロを含んでいる）を要する確率P(n)は、P(n) ＝ (1-0.5-ｎ)である。１ワード当たりのフィールド・パーティションの数をｍとする。その場合には、要求されるループの繰り返し数がｎ以下である確率は、$(P(n))^m = (1-0.5^{-n})^m$である。図９は、一実施の形態による、前記方程式の様々な値を含んでいる表９００を示す。図のように、２～４回を超えたループのアンローリングは、適切である。

【０１５１】

コーデック

一実施の形態では、図１のコーダ部１０１とデコーダ部１０３との少なくとも一方は、可変モジュールを具える。本明細書のコンテキストでは、モジュールは、コーデック構成１００によって用いられる圧縮アルゴリズムと関連した確率分布曲線の急勾配を反映する。例えば、モジュールは、確率分布の負の指数を有する。モジュールは、所望のパラメータの関数に従って変化するが、一実施の形態では、モジュールが、データの前回の一式のコンテキストに依存し、ここでは、データのそのような一式が、コーデック構成１００の各種のモジュールによって処理されるビットの一式を引用する。さらに、モジュールは、実行長の関数に従う増加を回避する（すなわち、シーケンスで複数の同一ビット）。

【０１５２】

したがって、動的単調（DM：dynamic-monotonic）コーデック構成を設けることができる。モジュールがデータの前回の一式のコンテキストに依存し、モジュールが実行長などの関数に従う増加を回避する、選択的な方法に関する更なる情報を、以下でより詳しく説明する。

【０１５３】

図１０は、この動的単調（DM）コーデック・構成及び他のアルゴリズム（例えば、Huffman、Rice Golomb、arithmetic等）の関係を示す、パフォーマンスレベルに対する計算の複雑性のレベルグラフ１０００である。図のように、DMコーデック構成は、予め設定されたパフォーマンスレベルを与えた最小限の計算の複雑性を用いるように設計される。

【０１５４】

ここで、図１のコーデック構成のコンテキストで実行される様々なオプション機能に関する更なる情報を説明する。一実施の形態では、DMコーデックが、状態スペース及びそのアップデート機能を記述することによって指定される（背景部分参照）。

【０１５５】

各状態は、ポジションP、context, shift, Aregister及びCregistaの五つの要素を有する。先に述べたように、モジュールは、データの前回の一式のコンテキストに基づき変化する。「コンテキスト」の上記定義を満足させる具体的な実施例によれば、本コンテキストは、入力アルファベットを超える長さkのビットストリング（合計2^k状態）を有する。Aregister及びCregisterはそれぞれ１以下である、2^{-n}の負でない倍数を維持する（各々合計2^n状態）。＜start＞状態及び＜finish＞状態の両方において、アレジスタとクレジスタとはともにゼロの値を有する。任意ではあるが、＜start＞状態用コンテキスト値initialは、エンコーダ及びデコーダの両方にとって同じように必要である。P値は、＜start＞状態では開始であり、＜finish＞状態では終了である。シフト値は、＜start＞と＜finish＞との状態では無関係である。

【０１５６】

アップデート関数φの具体例の一環として、いくつかの特定の固定関数がある。表１１参照。

【０１５７】

表１１

１．　関数mpsは、各コンテキスト値を入力アルファベットIの値に位置付ける。コンテキストで与えられた、より起こり得る記号であるIの記号であることを意図する。

２．　関数deltaは、2^k コンテキスト値のそれぞれを2^{-m}（

$0<m\leq n$

）に位置付ける。関数deltaの意図は、次の記号の値の確率についての情報を量的に捕らえることである。

【０１５８】

これら二つの関数は、DM制約に従って選択される。確率情報とともにこれら制約は、mps及びdeltaを選択する際に用いられるとともに、重要なアプリケーションにおいて、アルゴリズムの容易とエントロピコード化の効率の有用なコンビネーションを有効にする。

【０１５９】

一実施の形態では、DM制約は、表１２に示すものがある。

【０１６０】

表１２

１）動的制約：

delta(context) = 2^{-m}

$0<m\leq n$

ここで、m及びnは整数であり、deltaは、２の累乗の負の積分（negative integral power of two）である。次の記号がmps(context)によって与えられるもであることを要しないという条件付き確率を、delta(context)が近似するということを表している。

$$1\text{-Prob}(a == mps(context) \mid context) \approx delta(context)$$

２）単調制約：

$$(1 - delta(context)) \leq (1 - delta((2*context \bmod 2^k)+mps(context)))$$

また、このような不等式の右辺は、前回読まれた記号が最も起こりそうであるということを与える最もあり得る記号を読む近似的な確率である。単調制約は、次の記号が最もあり得る確率を、更に起こり得る記号が減らさないという最もらしい状況を反映する。

【０１６１】

これら２つの制約は、有効エントロピコーデックである。

【０１６２】

図１１は、一実施の形態による、エンコーダとデコーダの両方に対する更新関数を表す遷移表を示す。各ラインは、状態のセットを表し、状態のセットは、属性列の状態の各々を満足するものである。各行は、許容しうる状態の区分を形成する。

【０１６３】

適切な行が初期状態に対して識別された後、行の右側の動作が実行される。利用される全ての値は初期状態の値であり、行内のそのような動作シーケンスは必ずしも問題でない。新たな状態の各構成要素は、独自の値を受け取る。ブランクの記載は、状態の構成要素が変化しないことを意味する。

【０１６４】

エンコーダに対して、列の「共通」群からの更新動作及び列の「エンコーダ」群からの更新動作が実行される。デコーダに対して、列の「共通」群及び「エンコーダ」群から動作が選択される。状態遷移表１１００の下側において、動作の正確な定義が行われる。

【０１６５】
ＤＭ状態の効果は、Aregisterが常に（Ｆ１３で）加算される最新のデルタの乗算となることである。２進状態によって、デルタの２値表示が正確に一つの１ビットを有するようになる。単一状態によって、デルタは、コードシンボルが生成されるまで大きくならず、その結果、デルタ中のビットは、同一位置にのみ存在したままであり又は右に移動する。コードシンボルが生成されるまでこの状態のままであり、このポイントにおいて、Aregisterは零となる（。その理由は、最新のデルタの右側に対するAregisterビットのみが保持されるからである。）
【０１６６】
その結果、表１３の以下のセットの特性が付与される。
表１３
ａ）Ａを複数のデルタとする。
ｂ）コードシンボルを書き込んだ後にＡを零にする。
ｃ）コードシンボルを書き込んだ後の再正規化は必要ない。
【０１６７】
この同一の理由に対して、一般的な算術符号化法に要求されるCregisiterは、必ずしもＤＭエンコーダに用いられない。コードシンボル生成直後に、先行する状態シーケンスの全メモリがコンテキストで取得される。Contextが以前のｋ入力シンボルであるので、ＤＭコーデックは、短期記憶を有し、その結果、迅速に入力ストリングの局所統計に適合する。
【０１６８】
現存する構成の種々のアプリケーションが存在する。例えば、有効関数のコンテキストにおいて、画像、ビデオ及び信号処理は、しばしば、信号を「終結する」(concentrate)、すなわち、少数の重要な係数及び（零に置き換えることによって捨てられる）多数の無視できる係数を生成することを目的とする変換を伴う。無視できない係数の識別（又は位置）は、通常、その値と同様に重要である。この情報は、しばしば、無視できない係数を“１”にマップ付けするとともに無視できる係数を“０”にマップ付けする「有効関数」(significance function)で取得される。
【０１６９】
係数及びその有効ビットを適切な順番でリストすることによって、有効ビットを、その直前から良好な精度で予測することができる。その順番が、予測される大きさが小さくなる順番で係数をリストする場合、優勢的に(predominantly)１で開始するとともに優勢的に０で終了する有効ビットストリングを取得することができる。ストリングが進行すると統計が変化するそのようなストリングは、「非定常的」と称される。そのようなストリングの有効なエントロピ符号化は、直前の文脈のメモリを必要とすることがある。このメモリは、良好な予測精度を達成するのに十分大きく、十分急速に適合するのに十分短い(short-lived)。
【０１７０】
非定常的な実行の符号化のコンテキストにおいて、上記規定によれば、有効関数内の実行は、ビットのサブストリングを有し、この場合、最終ビット以外の全てのビットが一方の値を有し、最終ビットが他方のビットを有する。以前の実行の最終ビットの直後に次の実行を開始する。
【０１７１】
一般的な規則として、コンテキスト中のビットが多くなる（ｋが大きくなる）と、符号化速度が制限エントロピに更に近づく。しかしながら、ｋが大きくなると、実現が更に高価なものになる。ｋの範囲は、ｋの値の各々を検査できるように十分制限される。一度ｋが選択されると、非定常的な実行に対するＤＭコーデックのパフォーマンスは、mps(context)関数及びdelta(context)関数で全体的に取得される。これら関数を、表１４に従うような実験データから近似することができる。
表１４

1) $mps(context) = 1 \text{ if } \mathrm{Prob}(\text{next symbol} = 1 \mid context) \geq 0.5$

2) $\mathrm{Prob}(2^{2*delta(context)} \mid context) = 0.5$

【０１７２】
この手順は、コンテキストによって適合された十分な実験データを収集し、各コンテキストに対して、ヒストグラムを形成する。これから、確率関数を近似することができる。関数mps(context)を直接計算することができる。さらに、関数delta(context)を、表１４の２）の反復開放によって計算することができる。

【０１７３】
したがって、ＤＭコーデックは、入力ストリングを複数符号化ストリング及び単一符号化ストリングに１：１でマップ付けし、復号の再に、元の入力ストリングが生じる。一部の入力ストリングの符号化の際に、必ずしも全ての出力ストリングを発生させなくてよい。一部の入力ストリングは、比較的短い符号化ストリングに符号化することができ、多くの入力ストリングは、比較的長い符号化ストリングに符号化することができる。符号化する符号化ストリングに対する入力ストリングの長さに関して、種々のあり得る入力ストリングの発生の確率を説明するのが有益である。コーデックが、有用な圧縮特性を有する場合、短いストリングに符号化する入力ストリングの発生の確率は、長いストリングに符号化する入力ストリングの発生の確率より著しく大きくなる。

【０１７４】
動的確率を適用する必要はない。エントロピ符号化アプリケーションに対して、有効ビットストリームの統計は、変化することができるとともに、しばしば急速に変化することができ、かつ、変化するとともに、しばしば急速に変化する。そのような変化を適用確率テーブルによって追跡する必要はなく、多数の実行に亘っても低速にしか変化しない。したがって、ＤＭコーダは、必ずしも確率テーブルを使用せず、最後の少数ビット内又は単一の実行内で適合する。

【０１７５】
有効ビットデータによる実験的なテストは、contextの大抵の利益が有効ビットストリングの最後の少数ビットのみによって取得されることを表す。contextとしての有効ストリングスのこれら最後の少数ビットは、次のビットの確率を調整するのに用いられる。重要な確率の量は、$P_{context}$=Prob(next input bit=LSB|context)である。ＬＳＢの規定により、$P_{context}$<0.5となる。次のビットを最小で表すように追加することができるエントロピは、表１５のようになる。

表１５

$$entropy = -((1-p_{context})*\log_2(1-p_{context}) + p_{context} * \log_2(p_{context}))\ \text{bits}$$

【０１７６】
この場合、delta(context)=entropy/2となる。その理由は、Aregisterがスケール化されて２-1ビットを出力するからである。$P_{context}$～0.5の場合、次のビットに対して追加することができるエントロピは、近似的に表１６に示すようになる。

表１６

$$-((1-0.5)*\log_2(1-0.5) + 0.5 * \log_2(0.5)) \sim 1\ \text{bit}$$

【０１７７】
さらに、delta(context)=1/2となる。$P_{context}$<<0.5の場合、次のビットに対して追加することができるエントロピは、近似的に表１７に示すようになる。

表１７

$(1-p_{context})*(-p_{context}) - p_{context} * \log_2(p_{context}) \sim p_{context} * \log_2(1/p_{context})$

さらに、delta(context)=$P_{context}e*\log_2(1/P_{context})/2$となる。表１４の式２）参照。

【０１７８】

視覚的な見地

図１２は、一実施の形態によるクロミナンスの一時的な割合の減少を伴ってデータを圧縮する方法１２００を示す。一実施の形態において、この方法１２００を、図１の変換モジュール１０２のコンテキスト及び可逆変換を行うように実施することができる。しかしながら、方法１２００を任意の所望のコンテキストで実現することができる。

【０１７９】

処理１２０２において、フレームのluminescence(luma)データが、予め設定された第１のレートで更新される。処理１２０４において、フレームのluminescence(luma)データが、予め設定された第１のレート未満の予め設定された第２のレートで更新される。

【０１８０】

したがって、デジタルビデオ圧縮システムにおいて、シーンの互いに相違する構成要素の時間的なディテール(temporal detail)を送信する実効レートを変更することができる。例えば、変換された信号のある成分が他の成分に比べて頻繁に送信されるようにデータストリームを配置することができる。この一例において、ビデオシーケンスの３次元（空間＋時間）ウェーブレット(wavelet)変換を演算するとともに、結果的に得られるluma係数を全フレームレートで送信することができる。

【０１８１】

さらに、クロミナンス信号から一つ以上の高周波帯域を省略することができ、したがって、実際には、クロミナンス情報に対するビデオの一時的な応答速度－一時的なディテール忠実度を低減する。観察用に圧縮ビデオの再構成中、情報が送信されなかった「零レベル」を示すのではなく、省略された情報を近似によって充填し又は補間することができる。これを、省略された空間的なディテールに対するのと同様にして行うことができる。最も簡単には、それを、新たな情報を受信するまで最近受信したレベルを保持することによって行うことができる。さらに一般的には、それを、零又は省略された情報に対する他のデフォールト値を用いて逆ウェーブレットフィルタを算出して行って、減少した一時的なディテール以外の適切な全体に亘るレベルをゆする円滑な変化を生じることができる。

【０１８２】

この種のクロミナンスレート圧縮のある特定例は、以下の通りである。クロミナンス成分に対して、空間的に変換されたクロミナンス値の２フレーム（４フィールド）に亘る平均を算出することができる。これを、ダブルハールウェーブレットフィルタ対(double Haar wavelet filter pair)を適用するとともに最低周波数成分を除く全てを除くことによって達成することができる。この平均値のみを送信することができる。再構成に際し、クロミナンスの２フレーム（４フィールド）に亘って受信した値を保持することができる。観察者は、欠陥に対して圧縮方法を臨界的に試験するときでもこれに気が付かない。

【０１８３】

ビデオ圧縮処理、量子化、及び同様な値を互いにグループ分けするとともに対応する値(representative value)のみを送信することによる情報の除去の次の段階を示す。これは、正確に領域を明るくする方法及び正確な色のディテールを除外する。変換された成分が零に近いことを見つけると、対応する値に対して零を選択する（特定のスケールの変化がないことを示す。）。零の送信を省略することができ、デフォールト値として零と仮定する受信機を有する。これは、送信されるデータの量を低減することによって圧縮を助ける。レベルに対する人間の視覚は、ルミネッセンスとクロミナンスとの間で互いに相違することが知られている。

【０１８４】
したがって、ルミネッセンス成分及びクロミナンス成分に対して互いに相違するレベルを適用するとともにクロミナンスから更に情報を除くことによって、この事実の利点を取得することができる。このような互いに相違する量子化が、時間的な又は３次元の変換に従って行われると、空間的なディテールに従ってクロミナンスバンドの時間的なディテールが減少するという効果が生じる。典型的なケースにおいて、通常のビデオマテリアルに対して、時間的な変換の結果、高周波成分より著しく大きい低周波成分となる。このように変換された結果に量子化を適用すると、零を有する小さい値のグループとなり、実際には、圧縮表示から省略したものを収集し、クロミナンス成分の時間的な解像度を低下する。

【０１８５】
図１２Ａは、一実施の形態による、再生中に高品質の停止機能でデータを圧縮する方法１２５０を示す。一実施の形態において、この方法１２５０を、図１の構成のコンテキストで実施することができる。しかしながら、方法１２５０を、任意の所望のコンテキストで実現することができる。

【０１８６】
処理１２５２において、ビデオデータが圧縮される。一実施の形態において、データ圧縮を、図１の構成のコーダ部１０１のコンテキストで実行することができる。当然、そのような圧縮を、任意の所望のコンテキストで実現することができる。

【０１８７】
処理１２５４において、休止情報が圧縮データに挿入される。一実施の形態において、休止情報を用いて、再生ビデオデータの品質を向上することができる。さらに、休止情報は、高解像度フレームを有することができる。また、休止情報は、高解像度フレームを構成するのに用いられることができるデータを有することができる。

【０１８８】
したがって、処理１２５６において、再生中にビデオデータを休止する際に休止情報を用いることができる。この方法において、圧縮ビデオデータには、特にビデオが休止されるときに用いられる特別の情報のセットが設けられる。この特別な情報は、高品質フレーム、すなわち、規則的な圧縮フレームと組み合わせる際に高品質のフレームとなる差分情報を有することができる。

【０１８９】
圧縮ビットレートを有用なレベルに保持するために、この特別な情報を各フレームに含ませる必要がなく、一部のフレームのみに含ませればよい。典型的には、特別な情報を、画像中の１５フレーム前後ごとの１フレームに含ませることができ、これによって、高品質の休止動作を、１／２秒の時間精度で発生することができる。これを、ビデオ休止動作の観察にしたがって行うことができる。しかしながら、ビットレートを犠牲にするとこれより多くの特別な情報を有することができる。ユーザの便宜を犠牲にすると、更に良好な圧縮パフォーマンスを取得するためにしばしば更に少ない特別な情報を有することができる。これらの相反することを、２フレームから６０以上のフレームまでの範囲に亘って行うことができる。

【０１９０】
一実施の形態において、特別な情報は、互いに相違するパラメータセット（例えば、更に少ない情報とは別の量子化）を用いて又は互いに相違する圧縮方法（例えば、ＭＰＥＧストリーム内のＪＰＥＧ－２０００）を用いて圧縮されたビデオの全フレームを有することができる。これら特別なフレームを、元のビデオが圧縮される際に算出することができ、送信され又は格納された圧縮ビデオの規則的な圧縮ビデオフレームに従って搬送することができる。

【０１９１】
他の実施の形態において、特別な情報は、規則的な圧縮解除プロセス用の特別な情報を有することができ、それは、完全な特別のフレームではない。例えば、ウェーブレットビ

デオ圧縮機において、特別な情報は、通常の圧縮で除かれるが休止の際に特別の視覚的な急峻に対して保持されるデータのフィルタ帯域からなる。他の例において、特別な情報は、変換された係数からの特別な低次ビットの情報と、選択された休止可能なフレームに対する更に小さい量子化セッティングの使用に起因する他の係数とを有することができる。

【０１９２】

他の実施の形態において、特別な情報は、規則的な圧縮解除プロセスとは異なる圧縮解除プロセスで用いられるデータを有することができ、それは、完全なフレームではない。この情報は、圧縮解除後に、規則的なプロセスによって圧縮解除されたビデオの一つ以上のフレームと組み合わせて、更に詳しい静止フレームを生成することができる。

【０１９３】

ウェーブレットに基づく変換の例に関する更なる情報をここで説明し、それを、図１及び１２Ａの種々の形態と組み合わせて用いることができる。しかしながら、そのようなウェーブレットに基づく変換は、図示のためにのみ説明され、任意の方法に制約するものとして構成するものでない。例えば、図１及び１２Ａの種々の形態を、ＤＣＴに基づくアルゴリズムなどのコンテキストで実現できる。

【０１９４】

図１３は、一実施の形態に従ってデータを圧縮／圧縮解除する方法１３００を示す。一実施の形態において、この方法１３００を、図１の変換モジュール１０２のコンテキストで可逆変換を行うように実行することができる。しかしながら、方法１３００を、任意の所望のコンテキストで実現することができる。

【０１９５】

処理１３０２において、データを圧縮するために補間式が受信される（すなわち、それが識別され、メモリから探索される、等々。）。本明細書のコンテキストにおいて、データは、圧縮されることができる任意のデータを言及することができる。さらに、補間式は、補間を用いる任意の式（すなわち、ウェーブレットフィルタなど）を有することができる。

【０１９６】

処理１３０４において、少なくとも一つのデータ値が補間式によって要求され、この場合、要求されたデータ値は利用できない。そのようなデータ値は、上記データの任意のサブセットを有することができる。利用できないことによって、必要なデータ値は、存在せず、範囲外である、等々。

【０１９７】

その後、要求された利用できないデータ値を発生するために、補間処理を実行する。処理１３０６参照。補間式は、補間を用いる任意の式を有することができる。この形態によって、データの圧縮が強められる。

【０１９８】

図１４は、方法１３００が実行されるデータ構造１４００を示す。示されたように、変換中に、「ベストフィット１４０１」を、複数のデータ値１４０２に関連する補間式１４０３によって達成することができる。図１３の方法１３００の処理１３０２参照。データ値１４０２の一つが利用できない（例えば、１４０４）ことを決定した場合、補間式を用いて、そのような利用できないデータ値を発生することができる。上記技術の実現の一例に関する更なる詳細を、図１５の参照中に更に詳しく説明する。

【０１９９】

図１５は、一実施の形態によるデータを圧縮／圧縮解除する方法１５００を示す。オプションとして、この方法１５００を、図１の変換モジュールのコンテキストで実行するとともに、可逆変換を行うように実行する。しかしながら、方法１５００を、任意の所望のコンテキストで実現することができる。

【０２００】

方法１５００は、ウェーブレットフィルタ対に対してエッジフィルタを生じる技術を提供する。先ず、処理１５０２において、ウェーブレットフィルタが近似する局所導関数(l

ocal derivative)を決定するために、ウェーブレット形態を分析する。次に、処理１５０４において、多項式の次数が、補間に用いるためにウェーブレットフィルタの特性及び複数の利用できるサンプルに基づいて選択される。次に、補間式が、選択された多項式の次数を用いて各ウェーブレットフィルタに対して取得される。処理１５０６参照。さらに、処理処理１５０８において、特別なエッジウェーブレットのケースが、各ケースの利用できるサンプルを有する補間式を用いて取得される。

【０２０１】

さらに、補間式の一例に関する他の情報及び関連の情報を、詳細に説明する。

ＪＰＥＧ２０００規格１）で指定された変換の一つは、式＃１．１及び１．２に示す可逆５－３変換である。

式＃１．１及び１．２

$$Y_{2n+1} = X_{2n+1} - \left\lfloor \frac{X_{2n} + X_{2n+2}}{2} \right\rfloor \qquad eq\ 1.1$$

$$Y_{2n} = X_{2n} + \left\lfloor \frac{Y_{2n-1} + Y_{2n+1} + 2}{4} \right\rfloor \qquad eq\ 1.2$$

【０２０２】

Y_{2N-1}を左から近似するために、２次の多項式を左から適合することができる。利用できる値を用いた２Ｎ－１での２次導関数の半分の負の近似によって、式＃１．１．Ｒが生じる。

式＃１．１Ｒ

$$Y_{2N-1} = -\frac{1}{3}\left(X_{2N-1} - \left\lfloor \frac{3X_{2N-2} - X_{2N-4} + 1}{2} \right\rfloor \right) \qquad eq\ 1.1.R$$

【０２０３】

．１が最も右にあるときに、式＃１．１．Ｒを式＃１．１の代わりに用いることができる。明らかな３の乗算を、桁上り及び加算によって行うことができる。３による除算は更に複雑である。最も右にある指標が２Ｎ－１であるこの場合に対して、式＃１．２によるY_{2N-1}の計算に問題はない。最も右の点の指標が偶数（例えば２Ｎ）の場合において、式＃１．１に問題がないが、式＃１．２は、値の消失を伴う。ここでは、この点の場合において以前に計算された奇数の指標のY_S，Y_1及びY_3を用いて、Yの概算を偶数Xから除算することを目的とする。指標２Ｎでこのように要求される概算を、既に説明したような線形的な補間によって取得することができる。

式＃１．２．Ｒ

$$Y_{2N} = X_{2N} + \left\lfloor \frac{3Y_{2N-1} - Y_{2N-3} + 2}{4} \right\rfloor \qquad eq\ 1.2.R$$

【０２０４】

対応する状況は、左側の境界に当てはまる。同様なエッジフィルタが、左からではなく右（内側）からの必要とされる補間とともに適用される。この場合において、適切なフィルタが、式＃１．１．Ｌ及び１．２Ｌによって表される。

式＃１．１．Ｌ及び１．２．Ｌ

$$Y_0 = -\frac{1}{3}\left(X_0 - \left\lfloor \frac{3X_1 - X_3 + 1}{2} \right\rfloor \right) \qquad eq\ 1.1.L$$

$$Y_0 = X_0 + \left\lfloor \frac{3Y_1 - Y_3 + 2}{4} \right\rfloor \qquad eq\ 1.2.L$$

【０２０５】

逆変換フィルタを、元のフィルタに対する場合のように、すなわち、後退代入(back substitution)によって補間境界フィルタに対して取得することができる。逆変換境界フィルタを、順境界フィルタ(forward boundary filter)が用いられるのと全く同一の環境で標準的なフィルタの代わりに用いることができる。そのようなフィルタは、式＃２．１．Ｒ．ｉｎｖ，２．２．Ｒ．ｉｎｖ，２．１．Ｌ．ｉｎｖ及び２．２．Ｌ．ｉｎｖによって表される。

式＃２．１．Ｒ．ｉｎｖ，２．２．Ｒ．ｉｎｖ，２．１．Ｌ．ｉｎｖ及び２．２．Ｌ．ｉｎｖ

$$X_{2N-1} = -3Y_{2N-1} + \left\lfloor \frac{3X_{2N-2} - X_{2N-4} + 1}{2} \right\rfloor \qquad eq\ 2.1.R\quad inv$$

$$X_{2N} = Y_{2N} - \left\lfloor \frac{3Y_{2N-1} - Y_{2N-3} + 2}{4} \right\rfloor \qquad eq\ 2.2.R\quad inv$$

$$X_0 = -3Y_0 + \left\lfloor \frac{3X_1 - X_3 + 1}{2} \right\rfloor \qquad eq\ 2.1.L\quad inv$$

$$X_0 = Y_0 - \left\lfloor \frac{3Y_1 - Y_3 + 2}{4} \right\rfloor \qquad eq\ 2.2.L\quad inv$$

【０２０６】

したがって、一実施の形態は、フィルタの視覚的な特性を保持しながら従来の追加のステップを回避する５－３フィルタの再公式化を利用することができる。

式＃３．１，３．１Ｒ，３．２，３．２Ｌ

$$Y_{2n+1} = \left(X_{2n+1} + 1/2\right) - \left\lfloor \frac{\left(X_{2n} + 1/2\right) + \left(X_{2n+2} + 1/2\right)}{2} \right\rfloor \qquad eq\ 3.1$$

$$Y_{2N+1} = \left(X_{2N+1} + 1/2\right) - \left(X_{2N} + 1/2\right) \qquad eq\ 3.1R$$

$$\left(Y_{2n} + 1/2\right) = \left(X_{2n} + 1/2\right) + \left\lfloor \frac{Y_{2n-1} + Y_{2n+1}}{4} \right\rfloor \qquad eq\ 3.2$$

$$\left(Y_0 + 1/2\right) = \left(X_0 + 1/2\right) + \left\lfloor \frac{Y_1}{2} \right\rfloor \qquad eq\ 3.2L$$

【０２０７】

そのような公式において、所定の係数が、１／２のオフセット又はバイアスによって算出される。この公式において１／２を複数回加算しているように見えるが、これらの加算は、実際には演算中に生じる必要がない。式＃３．１及び３．１Ｒにおいて、１／２の加算の影響が相殺されているのを見ることができ、その結果、これらを入力データに適用する必要がない。その代わりに、括弧内の項(Yo+1/2)などを、係数として実際に計算され及び格納される量の名称と理解すべきであり、ウェーブレット変換ピラミッド(wavelet transform pyramid)の以下のレベルにされる。

【０２０８】

上記ケースのように、ＪＰＥＧ－２０００の逆フィルタを、以下の式＃４．２，４．２Ｌ，４．１，４．１Ｒで再公式化される。

式＃４．２，４．２Ｌ，４．１，４．１Ｒ

$$(X_{2n}+1/2)=(Y_{2n}+1/2)-\left\lfloor\frac{Y_{2n-1}+Y_{2n+1}}{4}\right\rfloor \qquad eq\ 4.2$$

$$(X_0+1/2)=(Y_0+1/2)-\left\lfloor\frac{Y_1}{2}\right\rfloor \qquad eq\ 4.2L$$

$$(X_{2n+1}+1/2)=Y_{2n+1}+\left\lfloor\frac{(X_{2n}+1/2)+(X_{2n+2}+1/2)}{2}\right\rfloor \qquad eq\ 4.1$$

$$(X_{2N+1}+1/2)=Y_{2N+1}+(X_{2N}+1/2) \qquad eq\ 4.1R$$

【０２０９】
ここでわかるように、逆演算に対する入力として取り出される値は、式＃３．１～３．２Ｌの順演算によって生成した同一項であり、明らかに、１／２による補正を計算する必要がない。
【０２１０】
このようにして、ウェーブレット変換の演算中に実行される演算処理の総数が減少される。
【０２１１】
本発明は、上記実施の形態に限定されるものではなく、幾多の変更及び変形が可能である。
【図面の簡単な説明】
【０２１２】
【図１】一実施の形態による、圧縮／圧縮解除データの構成を示す図である。
【図２】一実施の形態による、例外処理の方法を示す図である。
【図３】図２の方法の典型的な処理順序を示す図である。
【図４】他の実施の形態による、様々な処理特性に関連するグラフである。
【図５】他の実施の形態による、様々な処理特性に関連するグラフである。
【図６】他の実施の形態による、様々な処理特性に関連するグラフである。
【図７】他の実施の形態による、様々な処理特性に関連するグラフである。
【図８】他の実施の形態による、様々な処理特性に関連するグラフである。
【図９】他の実施の形態による、様々な処理特性に関連する表である。
【図１０】現在の二進法・単調（dyadic-monotonic）コーデック構成と、他のアルゴリズムとの関係を示す、パフォーマンスレベルに対する計算の複雑性のグラフである。
【図１１】一実施の形態による、エンコーダとデコーダとの間のアップデート関数（function）を示す転換表である。
【図１２】一実施の形態による、クロミナンス（クロマ）時間率減少でデータを圧縮する方法を示す図である。
【図１２Ａ】図１２Ａは、一実施の形態による、再生中に高品質停止能力でデータを圧縮する方法を示す図である。
【図１３】一実施の形態による、圧縮／圧縮解除データの方法を示す図である。
【図１４】図１３の方法が実行されるデータ構造を示す図である。
【図１５】一実施の形態による、圧縮／圧縮解除データの方法を示す図である。

【図１】

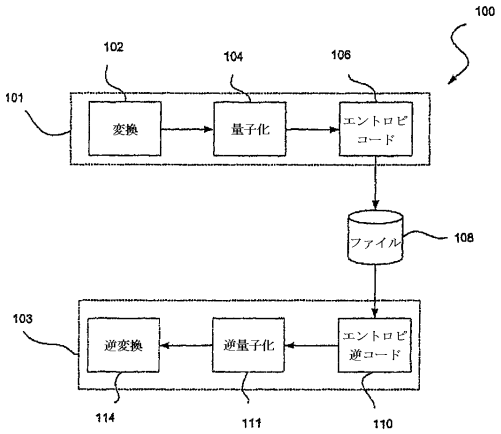


FIG. 1

【図２】

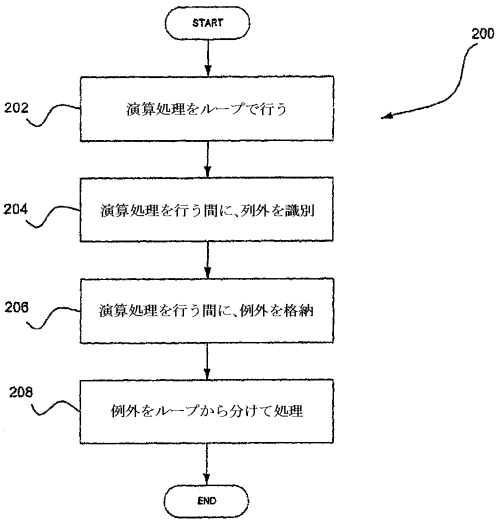


FIG. 2

【図３】

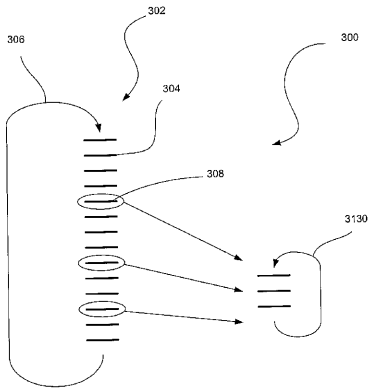


FIG. 3

【図４】

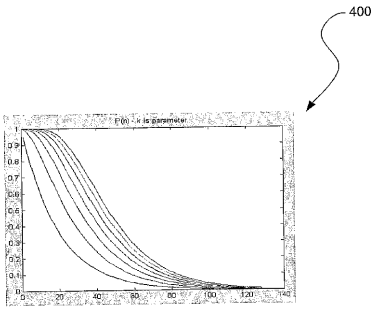


FIG. 4

【図５】

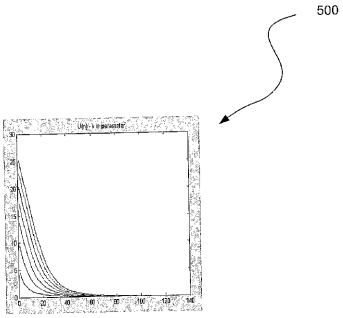


FIG. 5

【図６】

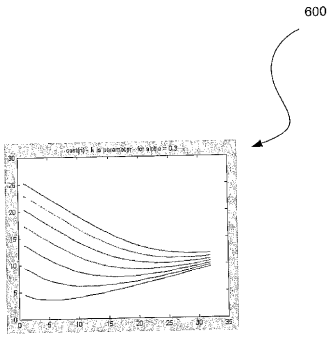


FIG. 6

【図７】

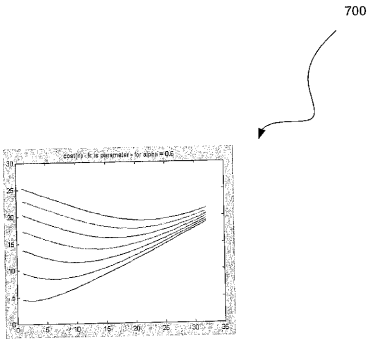


FIG. 7

【図８】

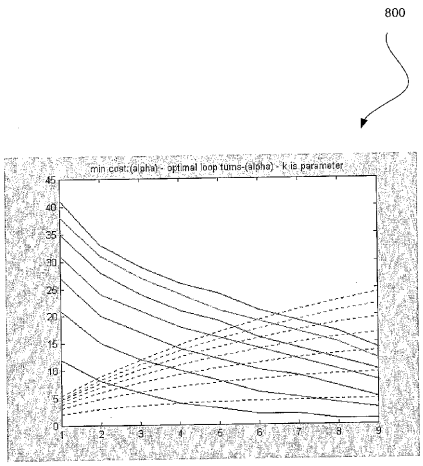


FIG. 8

【図９】

900

	n=1	n=2	n=3	n=4	n=5	n=6	n=7
m=2	0.25	0.56	0.77	0.88	0.94	0.97	0.99
m=4	0.06	0.32	0.59	0.77	0.88	0.94	0.98
m=8	0.00	0.10	0.34	0.60	0.78	0.88	0.97

FIG. 9

【図１０】

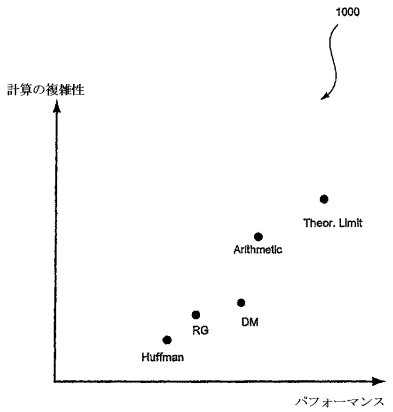


FIG. 10

【図１１】

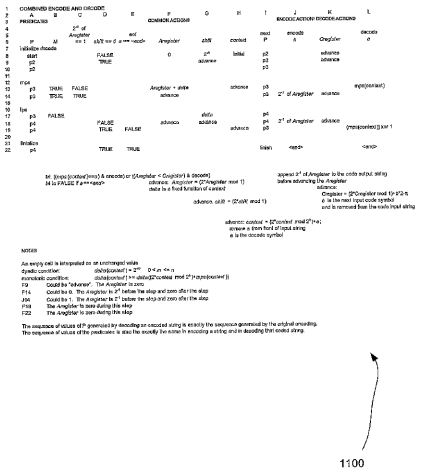


FIG. 11

【図１２】

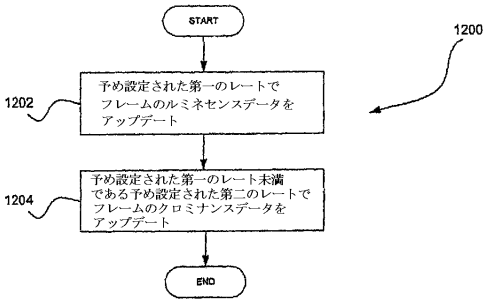


FIG. 12

【図１２Ａ】

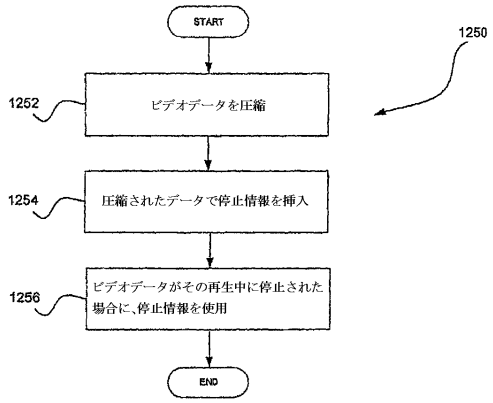


FIG. 12A

【図１３】

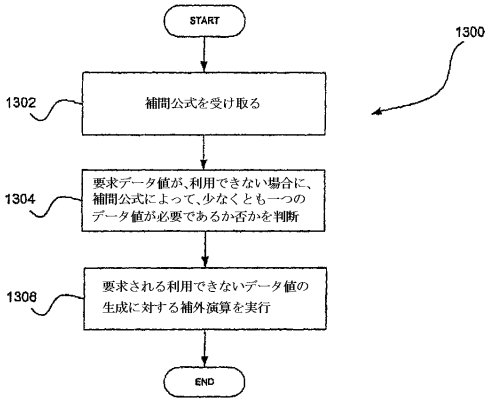


FIG. 13

【図１４】

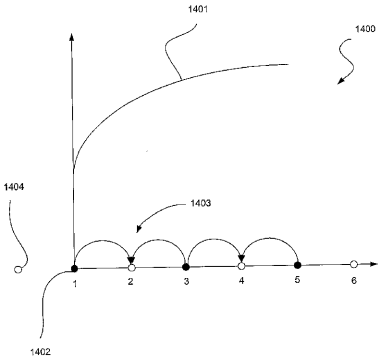


FIG. 14

【図１５】

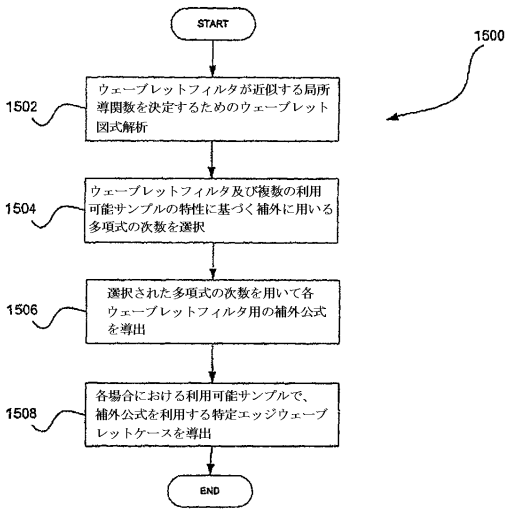


FIG. 15

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US03/16908

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) : G06F 17/14; G06F 12/00
US CL : 708/401; 711/125
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 708/401; 711/125

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

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

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5,893,145 A (THAYER ET AL.) 06 April 1999 (06.04.1999), column 6, lines 1-67; column 7, lines 1-67; column 8, lines 1-67; column 9, lines 1-67; column 10, lines 1-36	1-49
A	US 6,141,673 A (THAYER ET AL.) 31 October 2000 (31.10.2000), column 6, lines 1-67; column 7, lines 1-67; column 8, lines 1-67; column 9, lines 1-67; column 10, lines 1-45	1-49

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

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

"E" earlier application or patent published on or after the international filing date

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
29 July 2003 (29.07.2003)	29 SEP 2003
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US Commissioner for Patents P.O. Box 1450 Alexandria, Virginia 22313-1450 Facsimile No. (703)305-3230	Authorized officer Apu Mofiz for James R. Matthews Telephone No. 7036054240

Form PCT/ISA/210 (second sheet) (July 1998)

フロントページの続き

(31)優先権主張番号 60/390,492
(32)優先日 平成14年6月21日(2002.6.21)
(33)優先権主張国 米国(US)
(31)優先権主張番号 60/390,345
(32)優先日 平成14年6月21日(2002.6.21)
(33)優先権主張国 米国(US)

(81)指定国 AP(GH,GM,KE,LS,MW,MZ,SD,SL,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,MD,RU,TJ,TM),EP(AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,HU,IE,IT,LU,MC,NL,PT,RO,SE,SI,SK,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,ML,MR,NE,SN,TD,TG),AE,AG,AL,AM,AT,AU,AZ,BA,BB,BG,BR,BY,BZ,CA,CH,CN,CO,CR,CU,CZ,DE,DK,DM,DZ,EC,EE,ES,FI,GB,GD,GE,GH,GM,HR,HU,ID,IL,IN,IS,JP,KE,KG,KP,KR,KZ,LC,LK,LR,LS,LT,LU,LV,MA,MD,MG,MK,MN,MW,MX,MZ,NI,NO,NZ,OM,PH,PL,PT,RO,RU,SC,SD,SE,SG,SK,SL,TJ,TM,TN,TR,TT,TZ,UA,UG,UZ,VC,VN,YU,ZA,ZM,ZW

(74)代理人 100107227
弁理士 藤谷 史朗
(74)代理人 100114292
弁理士 来間 清志
(74)代理人 100119530
弁理士 冨田 和幸
(72)発明者 ウィリアム シー リンチ
アメリカ合衆国 カリフォルニア州 ９４３０３ パロ アルト トーマス ドライヴ ３３３１
(72)発明者 クラジミール ディー コラロヴ
アメリカ合衆国 カリフォルニア州 ９４０２５ メンロ パーク アヴィー アヴェニュー ２０５０
(72)発明者 スティーヴン イー サンダーズ
アメリカ合衆国 カリフォルニア州 ９５０１４ クパチーノ シャディーグローヴ ドライヴ ６０６９

Ｆターム(参考) 5B013 BB07 BB16
5B033 AA13 AA14 CA11
5B056 BB12 BB21 BB28 HH03