(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3977003号
(P3977003)
(45)発行日　平成19年9月19日(2007.9.19)　(24)登録日　平成19年6月29日(2007.6.29)

(51)Int.Cl.			F I		
G06F	17/14	(2006.01)	G06F	17/14	S
H03M	7/30	(2006.01)	H03M	7/30	A
H04N	7/30	(2006.01)	H04N	7/133	Z

請求項の数 2　(全 17 頁)

(21)出願番号　特願2000-297063 (P2000-297063)
(22)出願日　平成12年9月26日 (2000.9.26)
(65)公開番号　特開2002-108843 (P2002-108843A)
(43)公開日　平成14年4月12日 (2002.4.12)
審査請求日　平成16年9月10日 (2004.9.10)

(73)特許権者　503121103
株式会社ルネサステクノロジ
東京都千代田区大手町二丁目６番２号
(74)代理人　100089071
弁理士　玉村　静世
(74)代理人　100075096
弁理士　作田　康夫
(72)発明者　石川　誠
東京都国分寺市東恋ケ窪一丁目２８０番地　株式会社日立製作所中央研究所内
(72)発明者　海永　正博
東京都小平市上水本町五丁目２０番１号　株式会社日立製作所半導体グループ内

最終頁に続く

(54)【発明の名称】離散コサイン変換・逆離散コサイン変換手法及びその装置

(57)【特許請求の範囲】

【請求項１】

プロセッサと、N（N＝2^n　ただし、ｎは自然数）点離散コサイン変換処理を含むプログラムと、該離散コサイン変換処理を施すデータと、該離散コサイン変換処理のための行列データ及び係数データとを格納するための記憶装置とを含むデータ処理装置において、

該離散コサイン変換処理を、前後処理の追加により、使用する該プロセッサに合わせて2^k（ｋは自然数）回のN／2^k点離散コサイン変換処理に分割し、

該前後処理は、前記2^k（ｋは自然数）回のN／2^k点離散コサイン変換処理のための離散コサイン変換係数行列の少なくとも２つの係数行列が同一となるように施されるものであり、

該プロセッサは、レジスタファイルを有し、該離散コサイン変換処理を施す該データを該記憶装置から読み出し、該記憶装置に格納された該プログラムに基づいて、該前処理と、2^k（ｋは自然数）回の該N／2^k点離散コサイン変換処理と、該後処理とを行い、該N／2^k点離散コサイン変換処理の最初に該行列データ及び該係数データを該記憶装置から該レジスタファイルに１回だけロードして、該係数行列とし、該レジスタファイルに該係数行列を保持したまま、該N／2^k点離散コサイン変換処理を2^k（ｋは自然数）回繰返すことを特徴とするデータ処理装置。

【請求項２】

プロセッサと、N（N＝2^n（ｎは自然数））点逆離散コサイン変換処理を含むプログラムと、該逆離散コサイン変換処理を施すデータと、該逆離散コサイン変換処理のための行

列データ及び係数データとを格納するための記憶装置とを含むデータ処理装置において、
該逆離散コサイン変換処理を、前後処理の追加により、使用する該プロセッサに合わせて2^k（ｋは自然数）回のN／2^k点逆離散コサイン変換処理に分割し、
該前後処理は、前記2^k（ｋは自然数）回のN／2^k点逆離散コサイン変換処理のための逆離散コサイン変換係数行列の少なくとも２つの係数行列が同一となるように施されるものであり、
該プロセッサは、レジスタファイルを有し、該逆離散コサイン変換処理を施す該データを該記憶装置から読み出し、該記憶装置に格納された該プログラムに基づいて、該前処理と、2^k（ｋは自然数）回の該N／2^k点逆離散コサイン変換処理と、該後処理とを行い、該Ｎ／2^k点逆離散コサイン変換処理の最初に該行列データ及び該係数データを該記憶装置から該レジスタファイルに１回だけロードして、該係数行列とし、該レジスタファイルに該係数行列を保持したまま、該Ｎ／2^k点逆離散コサイン変換処理を2^k（ｋは自然数）回繰返すことを特徴とするデータ処理装置。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は、マイクロプロセッサやマイクロコンピュータ等のデータ変換処理装置に関わり、特に離散コサイン変換や逆離散コサイン変換などを行う画像処理及び音声処理応用データ処理装置に係わる。
【０００２】
【従来の技術】
ディジタル化された画像及び音声はデータ量が巨大であり、その蓄積や伝送の際に問題となる。従って、格納前に圧縮しておき、使用する際に伸長する、または、送信前に圧縮し、受信後に伸長するなどの対策が取られる。
【０００３】
以下、画像圧縮・伸長にを例に説明する。
【０００４】
圧縮
１．２次元離散コサイン変換
２．量子化
３．ハフマン符号化
伸長
４．ハフマン復号化
５．逆量子化
６．２次元逆離散コサイン変換
１の２次元離散コサイン変換及び６の２次元逆離散コサイン変換は８＊８画素の２次元ブロックを対象に行われ、変換結果も８＊８要素の２次元ブロック値群となる。
【０００５】
１の２次元離散コサイン変換変換により、８＊８ブロックの高周波成分に相当する要素は通常０に近い値が多くなり、重み付けされた量子化操作によってそれらの大多数は０となる。３のハフマン符号化は、８＊８のブロック要素群をビットストリームに変換する。この際、要素中に０が多いことを利用して変換するため、変換後のビットストリームの所用バイト数は１／１０程度になるといわれている。
【０００６】
伸長におけるハフマン複合化４は３と逆の操作を行い、ビットストリームから８＊８のブロック要素群を生成する。５の逆量子化は、２の量子化の際につけた重みの逆数を乗じることで、量子化前の要素群を復元する。６の２次元逆離散コサイン変換は１の逆の操作をすることで、８＊８画素を復元する。
【０００７】
ここで、上記の圧縮・伸長操作に必要な手間を示す。通常の汎用プロセッサを用いた場合

、８＊８ブロックあたり、圧縮・伸長ともにそれぞれ２０００～３０００命令必要である。６４０＊４８０のフルカラー（２４ビット／画素）画像を対象とする場合には１４４００倍となり、１静止画像あたり２８．８Ｍ～４３．２Ｍ命令の実行を要する。１命令の処理を１クロック、１００ＭＨｚで動作するプロセッサを用いた場合、２～４フレーム／秒の圧縮・伸長速度しか得られず、実時間で画像取り込み・画像再生を行うことで動画的な効果を得ることが難しいといえる。

【０００８】

そこで、画像及び音声圧縮・伸長を補助するための特殊な専用演算器を搭載するなどした処理装置を別途用意する方法が多く取られてきた。

【０００９】

また、もう一つの解決手段として汎用のプロセッサにも搭載されるようになった、行列・ベクトル演算命令や、複数の積和演算などを同時に処理するＳＩＭＤ（Ｓｉｎｇｌｅ Ｉｎｓｔｒｕｃｔｉｏｎ Ｍｕｌｔｉｐｌｅ Ｄａｔａ）方式の命令を利用して変換処理装置を構成する方法もあげられる。離散コサイン変換は線形変換であるため、Ｎ点の変換はＮ＊Ｎの行列で表現でき、上記の行列・ベクトル演算命令やＳＩＭＤ命令で行列の積を求めることで変換処理が完了する。

【００１０】

【発明が解決しようとする課題】

上記で示した２つ目の処理装置で使われる変換行列は、Ｎ点の離散コサイン変換にではＮ＊Ｎの大きさを持つことを述べた。例えば８＊８の２次元離散コサイン変換処理を繰り返して処理する場合には、図７に示す６４個の係数と８個の入力データを用いて８個の出力を得る行列演算を、縦方向８回、横方向８回繰り返すことになる。

【００１１】

このとき、６４個の係数は定数であるためプロセッサ内部に保持したまま処理することが理想的である。しかし、現実的にはレジスタ数の制約から主記憶に配置した６４個の一部を逐次読み込みながら処理することが要求される。

【００１２】

また、上記８点の変換処理を効率良く行なうためには、８＊８の行列演算器や、並列度８のＳＩＭＤ演算器を用意することが望ましいが、例えば３次元グラフィクス等の分野での利便性や回路面積を考慮すると、４＊４の行列演算や並列度４のＳＩＭＤ演算が合理的であるといえる。さらに、音声圧縮・伸長で利用される３２点の離散コサイン変換の場合は係数、回路の規模からより実現が難しくなる。

【００１３】

特開平９－２１２４８４の実施例において、図４に示すように８＊８の離散コサイン変換を偶関数部分と奇関数部分数に分解してすることで、８＊８行列の右上の４＊４成分、左下の４＊４成分が０になることが、結論に至る途中式に示されている。４＊４の行列と、４次元ベクトルの積を求める演算装置を備えたプロセッサを用いたとき、若干の前処理、後処理を追加することで２回の行列演算によって変換処理が可能であることが暗に示唆されている。

【００１４】

しかしながら、異なる４＊４の係数行列を２種類使用するため、離散コサイン変換処理で行列演算を行なう度に４＊４の係数データを入れ替える処理が必要となる。例えば、行列演算と係数入れ替えのサイクル数の比を１：１とした場合、行列演算器の利用効率は５０％以下になると概算できる。

【００１５】

本発明は、行列演算、ＳＩＭＤ演算による離散コサイン変換・逆離散コサイン変換の高速化を達成するために、上記変換係数行列の入れ替えにかかるオーバーヘッドを削減して演算器の利用効率を高めることを目的とする。

【００１６】

【課題を解決するための手段】

本発明では、上記の問題を解決するために、離散コサイン変換の定義式の性質を利用する。前処理のバタフライ演算と、後処理の加算処理を追加することで、Ｎ点の離散コサイン変換が、２つのＮ／２点の離散コサイン変換に分割できる。このとき、係数行列としてはただ１つの（Ｎ／２）＊（Ｎ／２）行列を使用するため、行列の入れ替えが不要となり、高速な変換処理が可能となる。

【００１７】

８点の変換処理における係数の数は、図７と比較すると１／４、図４と比較すると１／２となり、プロセッサ内に係数を保持するレジスタの数を低減できる。

【００１８】

この分割は任意の回数繰り返すことで、前後処理が若干増加するが、行列の係数、演算の規模が１／４、１／１６、．．．と縮小できるため、使用するプロセッサに最適な分割数を選択することができる。

【００１９】

逆離散コサイン変換に関しても上記の分割は成り立つため、同様に行列入れ替えを不要とした変換処理が可能となる。

【００２０】

【発明の実施の形態】

（１）離散コサイン変換の定義

１次元離散コサイン変換の定義式は図５のように示される。係数Ａの定義は式（３）であるが、画像圧縮のように離散コサイン変換後に量子化を行う場合、もしくは画像伸長のように逆離散コサイン変換前に逆量子化を行う場合には、その量子化時の重み係数にＡを乗じたものを利用すれば、離散コサイン変換の定義を式（３’）のように単純化できる。以後、１次元離散コサイン変換の定義式として式（２）、（３’）、（４）を利用することにする。

【００２１】

離散コサイン変換、逆離散コサイン変換の代表的な応用である静止画や動画の圧縮・伸長には、Ｎ＝８が利用される。そこで、図５の離散コサイン変換でＮ＝８とすれば、図７に示す行列演算で示すことができる。行列の要素は、図６に示すｃ（ｎ，Ｎ）の性質のうち、式（５）を利用して正規化している。逆離散コサイン変換は、この行列を転置した計算に相当するため、ここでは離散コサイン変換のみを扱う。

【００２２】

Ｎ＊Ｎ点の２次元離散コサイン変換は上記Ｎ点１次元離散コサイン変換を縦方向にＮ回、その後に横方向にＮ回（もしくは横方向にＮ回した後に縦方向にＮ回）行ったものとして定義できる。つまり、Ｎ＊Ｎ点の２次元離散コサイン変換は、Ｎ点の１次元離散コサイン変換２Ｎ回に分解できる。以後、１次元離散コサイン変換を中心に扱う。

【００２３】

（２）８点離散コサイン変換の分割

ここで、式（２）をｋ＝２Ｋ、ｋ＝２Ｋ＋１とおき、偶数出力、奇数出力に分離する。式（６）、式（７）の性質を利用すると、偶数側は式（９）のように、奇数側は式（１０）のように変形できる。また、奇数側でＫ＝Ｎ／２－１のとき、つまり、Ｘ［Ｎ－１］は式（６）の性質を利用すると、式（１１）のように導かれる。式（１０）、式（９）、式（１１）、にＮ＝８を代入してまとめると式（１）の行列表現が得られる。この行列の左上と右下の４＊４成分に注目すると、４点離散コサイン変換と同一の行列になっており、以上の式変形によって、８点離散コサイン変換はある前処理と後処理を追加することによって、２つの４点離散コサイン変換に分割できることを示すことができた。図２はそのバタフライ図を示したものである。

【００２４】

逆離散コサイン変換を行う場合には、演算の手順を出力側から入力側へ、逆順で行なう。その際、係数行列の逆行列が必要となるが、離散コサイン変換の係数行列の性質から、転置行列を用いればよい。以下で示す実施例では、離散コサイン変換のみを扱うが、ここで

述べた逆順に処理する方法を用いて、逆離散コサイン変換も実現できることを述べておく。

【００２５】

（３）実施例１

本実施例の変換装置の構成を図１に示す。変換装置１０１はプロセッサ１０２と記憶部１０３によって構成され、外部に接続された入力装置１１１からデータを入力し、画像の圧縮・伸長等の変換処理を行ない、出力装置１１２から変換結果を出力する。プロセッサ１０２と記憶部１０３はアドレスバス１０４とデータバス１０５によって接続され、プロセッサ内部のアドレス生成器１０６で計算されたアドレスによって記憶装置の番地を指定し、データバスを通じてプロセッサ内のレジスタファイル１０７とのデータ転送を行なう。記憶部１０３はプログラム記憶装置１０９とデータ記憶装置１１０から構成される。演算器１０８はレジスタファイルの内容を読み出して演算処理を行ない、再びレジスタファイルに結果を書き戻す。レジスタファイル１０７の構成を図１０に示す。Ｒ０～Ｒ１５で構成されるレジスタファイル１、ＸＲ０～ＸＲ１５で構成されるレジスタファイル２から構成され、レジスタファイル０のレジスタＲｎ、Ｒｎ＋１、Ｒｎ＋２、Ｒｎ＋３を組みにして、ＶＲｎと呼ぶことにする。

【００２６】

ここで、プロセッサ１０２の命令とその動作を定義する。まず、４＊４行列と要素数４のベクトルとの行列積を行うＴＲＶ命令を、以下のように記述するものとする。

【００２７】

ＴＲＶ　ＶＲｓ，ＶＲｄ（ｓ，ｄ＝４ｎ）

ＴＲＶ命令は、レジスタファイル１の１６本のレジスタを４＊４行列、レジスタファイル０内レジスタ群、ＶＲｓを４次元ベクトルとみなし、その行列とベクトルの乗算結果をＶＲｄへ格納する。図（１１）はＴＲＶ命令の動作内容を示したものである。

【００２８】

次に、加算、減算、乗算を行う命令として、ＡＤＤ、ＳＵＢ、ＭＵＬ、ＡＤＤ４、ＳＵＢ４、ＭＵＬ４を以下のように定義する。

【００２９】

ＡＤＤ　Ｒｓ，Ｒｔ，Ｒｄ
ＳＵＢ　Ｒｓ，Ｒｔ，Ｒｄ
ＭＵＬ　Ｒｓ，Ｒｔ，Ｒｄ
ＡＤＤ４　ＶＲｓ，ＶＲｔ，ＶＲｄ（ｓ，ｔ，ｄ＝４ｎ）
ＳＵＢ４　ＶＲｓ，ＶＲｔ，ＶＲｄ（ｓ，ｔ，ｄ＝４ｎ）
ＭＵＬ４　ＶＲｓ，ＶＲｔ，ＶＲｄ（ｓ，ｔ，ｄ＝４ｎ）

ＡＤＤ、ＳＵＢ、ＭＵＬ命令は、レジスタＲｓと、Ｒｔについて、加算、減算、乗算をし、結果をレジスタＲｄへ格納する。ＡＤＤ４、ＳＵＢ４、ＭＵＬ４命令は、レジスタ群ＶＲｓとのレジスタ群ＶＲｔの対応する要素について、加算、減算、乗算をし、結果をＶＲｄへ格納する。図（１２）、図（１３）、図（１４）はＡＤＤ４命令、ＳＵＢ４命令、ＭＵＬ４命令、の動作内容を示したものである。さらに、主メモリからレジスタにデータをロード、逆にレジスタから主メモリにデータをストアする命令として、以下の４命令を定義する。

【００３０】

ＬＤ　ｂ，ｄｉｓｐ，Ｒｄ
ＳＴ　ｂ，ｄｉｓｐ，Ｒｓ
ＬＤ４　ｂ，ｄｉｓｐ，ｓｔｅｐ，ＶＲｄ（ｄ＝４ｎ）
ＳＴ４　ｂ，ｄｉｓｐ，ｓｔｅｐ，ＶＲｓ（ｓ＝４ｎ）

ＬＤ命令は、主メモリのアドレス（ｂ＋ｄｉｓｐ）番地に格納されているデータをレジスタＲｄにロードする。ＳＴ命令は、レジスタＲｓの値を主メモリのアドレス（ｂ＋ｄｉｓｐ）番地にストアする。ＬＤ４命令は、主メモリのアドレス（ｂ＋ｄｉｓｐ）、（ｂ＋ｄｉｓｐ＋ｓｔｅｐ）、（ｂ＋ｄｉｓｐ＋２＊ｓｔｅｐ）、（ｂ＋ｄｉｓｐ＋３＊ｓｔｅｐ

）番地、に格納されているデータをレジスタファイル１内のレジスタＲｄ、Ｒｄ＋１、Ｒｄ＋２、Ｒｄ＋３にロードする。ＳＴ４命令は、レジスタＲｓ、Ｒｓ＋１、Ｒｓ＋２、Ｒｓ＋３の値を主メモリのアドレス（ｂ＋ｄｉｓｐ）、（ｂ＋ｄｉｓｐ＋ｓｔｅｐ）、（ｂ＋ｄｉｓｐ＋２＊ｓｔｅｐ）、（ｂ＋ｄｉｓｐ＋３＊ｓｔｅｐ）番地にストアする。

【００３１】

最後に、レジスタファイル０、レジスタファイル１の内容を入れ替えるＥＸＣＨＧ命令を定義する。

【００３２】

ＥＸＣＨＧ

以下に、上記の装置で８＊８の２次元離散コサイン変換を用いた画像変換処理を行なう例を示す。

【００３３】

図１の記憶装置１０３には、図１５に示すように、変換プログラム、４＊４の行列データ、４つの係数データ、８＊８画素＊Ｂブロックの画像データが主記憶上に格納されているとする。

【００３４】

まず、係数行列データとバタフライ演算係数データをレジスタにロードする。このロード作業は、Ｂブロックの離散コサイン変換処理の最初にただ１回だけ行えばよい。この操作により、レジスタファイルには図１０に示す係数がロードされ、これらの係数はＢブロック変換作業中に変更されない。

【００３５】

```
＃行列、係数ロード
ＬＤ４　　ＭＡＴＲＩＸ，０，１，Ｒ０
ＬＤ４　　ＭＡＴＲＩＸ，４，１，Ｒ４
ＬＤ４　　ＭＡＴＲＩＸ，８，１，Ｒ８
ＬＤ４　　ＭＡＴＲＩＸ，１２，１，Ｒ１２
ＥＸＣＨＧ
ＬＤ４　　ＣＯＥＦＦ，０，１，１２
```

次に８点１次元離散コサイン変換を行う命令列を示す。ＯＦＦはＩＭＧからのオフセットを示し第１回目の処理では０とする。

【００３６】

```
＃８点離散コサイン変換（横）
ＬＤ４　　ＩＭＧ，０＋ＯＦＦ，１，ＶＲ８
ＬＤ４　　ＩＭＧ，７＋ＯＦＦ，－１，ＶＲ４
ＡＤＤ４　ＶＲ８，ＶＲ４，ＶＲ０
ＳＵＢ４　ＶＲ８，ＶＲ４，ＶＲ４
ＭＵＬ４　ＶＲ４，ＶＲ１２，ＶＲ４
ＴＲＶ　　ＶＲ０，ＶＲ０
ＴＲＶ　　ＶＲ４，ＶＲ４
ＡＤＤ　　Ｒ４，Ｒ５，Ｒ４
ＡＤＤ　　Ｒ５，Ｒ６，Ｒ５
ＡＤＤ　　Ｒ６，Ｒ７，Ｒ６
ＳＴ４　　ＩＭＧ，０＋ＯＦＦ，２，ＶＲ０
ＳＴ４　　ＩＭＧ，１＋ＯＦＦ，２，ＶＲ４
```

ＯＦＦを８ずつ増加させながらこの１２命令で構成される変換処理を８回行なうことで、８＊８画素に対して横方向の１次元離散コサイン変換が完了する。その後、縦方向の変換を行うために、ＯＦＦを０、１、．．．、７と変化させながら以下の命令列を８回行なう。

【００３７】

```
＃８点離散コサイン変換（縦）
```

```
ＬＤ４　　ＩＭＧ，０＋ＯＦＦ，８，ＶＲ８
ＬＤ４　　ＩＭＧ，５６＋ＯＦＦ，－８，ＶＲ４
ＡＤＤ４　ＶＲ８，ＶＲ４，ＶＲ０
ＳＵＢ４　ＶＲ８，ＶＲ４，ＶＲ４
ＭＵＬ４　ＶＲ４，ＶＲ１２，ＶＲ４
ＴＲＶ　　ＶＲ０，ＶＲ０
ＴＲＶ　　ＶＲ４，ＶＲ４
ＡＤＤ　　Ｒ４，Ｒ５，Ｒ４
ＡＤＤ　　Ｒ５，Ｒ６，Ｒ５
ＡＤＤ　　Ｒ６，Ｒ７，Ｒ６
ＳＴ４　　ＩＭＧ，０＋ＯＦＦ，１６，ＶＲ０
ＳＴ４　　ＩＭＧ，５６＋ＯＦＦ，－１６，ＶＲ４
```

以上の操作により、８＊８の２次元離散コサイン変換を完了する。変換対象となるブロック数Ｂが十分大きいとすると、行列、係数ロードに必要な６命令を無視することができる。そのため、１ブロックあたり１９２命令で処理できるといえる。

【００３８】

従来例ではＴＲＶ命令毎に６命令の係数ロードが必要であり、１ブロックの変換中に３２回のＴＲＶ命令を使用することから、さらに６命令＊３２回＝１９２命令の追加となる。本発明により、命令数を半分に削減できたと言える。

【００３９】

（４）実施例２

本実施例は、行列演算命令ではなくベクトル内積演算命令を持つプロセッサを用いた場合の実装を示す。係数、データは実施例１で示した図１５のように主記憶上に配置されているとする。

【００４０】

本実施例で使用するプロセッサは、実施例１と以下の相違点を持つとする。

１．図１６に示す、３２本のレジスタから構成されるレジスタファイルだけを１つだけ持ち、そのためＥＸＣＨＧ命令は持たない

２．ＴＲＶ命令の代わりに、ＩＰＲ命令を持つ

ＩＰＲ　ＶＲｓ，ＶＲｔ，Ｒｄ（ｓ，ｔ＝４ｎ）

ＩＰＲ命令は、レジスタ群ＶＲｓと、ＶＲｔをそれぞれ４要素のベクトルとみなし、その内積をレジスタＲｄに格納する。図１７にその演算内容を示す。

【００４１】

以上のプロセッサを利用して、８＊８の離散コサイン変換、逆離散コサイン変換を行なう手順を以下に示す。

【００４２】

まず、行列データと係数データをレジスタにロードする。このロード作業は、Ｂブロックの離散コサイン変換処理の最初にただ１回だけ行えばよい。

【００４３】

```
＃行列、係数ロード
ＬＤ４　　ＣＯＥＦＦ，０，１，ＶＲ１２
ＬＤ４　　ＭＡＴＲＩＸ，０，４，ＶＲ１６
ＬＤ４　　ＭＡＴＲＩＸ，１，４，ＶＲ２０
ＬＤ４　　ＭＡＴＲＩＸ，２，４，ＶＲ２４
ＬＤ４　　ＭＡＴＲＩＸ，３，４，ＶＲ２８
```

次に８点１次元離散コサイン変換を行う命令列を示す。ＯＦＦはＩＭＧからのオフセットを示し第１回目の処理では０とする。

【００４４】

```
＃８点離散コサイン変換（横）
ＬＤ４　　ＩＭＧ，０＋ＯＦＦ，１，８
```

```
ＬＤ４　　ＩＭＧ，７＋ＯＦＦ，－１，４
ＡＤＤ４　ＶＲ８，ＶＲ４，ＶＲ０
ＳＵＢ４　ＶＲ８，ＶＲ４，ＶＲ４
ＭＵＬ４　ＶＲ４，ＶＲ１２，ＶＲ４
ＩＰＲ　　ＶＲ０，ＶＲ１６，Ｒ８
ＩＰＲ　　ＶＲ０，ＶＲ２０，Ｒ９
ＩＰＲ　　ＶＲ０，ＶＲ２４，Ｒ１０
ＩＰＲ　　ＶＲ０，ＶＲ２８，Ｒ１１
ＩＰＲ　　ＶＲ４，ＶＲ１６，Ｒ０
ＩＰＲ　　ＶＲ４，ＶＲ２０，Ｒ１
ＩＰＲ　　ＶＲ４，ＶＲ２４，Ｒ２
ＩＰＲ　　ＶＲ４，ＶＲ２８，Ｒ３
ＡＤＤ　　Ｒ１，Ｒ９，Ｒ１
ＡＤＤ　　Ｒ２，Ｒ１０，Ｒ２
ＡＤＤ　　Ｒ３，Ｒ１１，Ｒ３
ＳＴ４　　ＩＭＧ，０＋ＯＦＦ，２，ＶＲ８
ＳＴ４　　ＩＭＧ，１＋ＯＦＦ，２，ＶＲ０
```

ＯＦＦを８ずつ増加させながらこの１８命令で構成される変換処理を８回行なうこでで、８＊８画素に対して横方向の１次元離散コサイン変換が完了する。その後、縦方向の変換を行うために、ＯＦＦを０、１、．．．、７と変化させながら以下の命令列を８回行なう。

【００４５】

＃８点離散コサイン変換（縦）

```
ＬＤ４　　ＩＭＧ，０＋ＯＦＦ，８，８
ＬＤ４　　ＩＭＧ，５６＋ＯＦＦ，－８，４
ＡＤＤ４　ＶＲ８，ＶＲ４，Ｒ０
ＳＵＢ４　ＶＲ８，ＶＲ４，Ｒ４
ＭＵＬ４　ＶＲ４，ＶＲ１２，Ｒ４
ＩＰＲ　　ＶＲ０，ＶＲ１６，Ｒ８
ＩＰＲ　　ＶＲ０，ＶＲ２０，Ｒ９
ＩＰＲ　　ＶＲ０，ＶＲ２４，Ｒ１０
ＩＰＲ　　ＶＲ０，ＶＲ２８，Ｒ１１
ＩＰＲ　　ＶＲ４，ＶＲ１６，Ｒ０
ＩＰＲ　　ＶＲ４，ＶＲ２０，Ｒ１
ＩＰＲ　　ＶＲ４，ＶＲ２４，Ｒ２
ＩＰＲ　　ＶＲ４，ＶＲ２８，Ｒ３
ＡＤＤ　　Ｒ１，Ｒ９，Ｒ１
ＡＤＤ　　Ｒ２，Ｒ１０，Ｒ２
ＡＤＤ　　Ｒ３，Ｒ１１，Ｒ３
ＳＴ４　　ＩＭＧ，０＋ＯＦＦ，１６，ＶＲ８
ＳＴ４　　ＩＭＧ，５６＋ＯＦＦ，－１６，ＶＲ０
```

以上の操作により、８＊８の２次元離散コサイン変換を完了する。変換対象となるブロック数Ｂは十分大きいとすると、行列、係数ロードに必要な５命令を無視することができるため、１ブロックあたり２８８命令で２次元離散コサイン変換を処理できる。

【００４６】

（５）実施例３

本実施例は、行列演算命令ではなくＳＩＭＤ命令を持つプロセッサを用いた場合の実装を示す。係数、データは実施例１で示した図１５のように主記憶上に配置されているとする。

【００４７】

本実施例で使用するプロセッサは、実施例２と以下の相違点を持つとする。
１．ＩＰＲ命令の代わりに、ＭＡＣ４命令を持つ
２．ＭＵＬ４、ＭＡＣ４命令をブロードキャスト拡張した、ＭＵＬ４Ｂ、ＭＡＣ４Ｂ命令を持つ
ＭＡＣ４　ＶＲｓ，ＶＲｔ，ＶＲｄ（ｓ，ｔ，ｄ＝４ｎ）
ＭＡＣ４命令は、レジスタＲｓ、Ｒｓ＋１、Ｒｓ＋２、Ｒｓ＋３と、レジスタＲｔ、Ｒｔ＋１、Ｒｔ＋２、Ｒｔ＋３のそれぞれの積を、レジスタＲｄ、Ｒｄ＋１、Ｒｄ＋２、Ｒｄ＋３に足し込む。
【００４８】
ＭＵＬ４Ｂ　ＶＲｓ，ＶＲｔ，ＶＲｄ，ｂ（ｓ，ｔ，ｄ＝４ｎ、ｂ＝０～３）
ＭＡＣ４Ｂ　ＶＲｓ，ＶＲｔ，ＶＲｄ，ｂ（ｓ，ｔ，ｄ＝４ｎ、ｂ＝０～３）
ＭＵＬ４Ｂ命令は、レジスタＲｓ＋ｂ、Ｒｓ＋ｂ、Ｒｓ＋ｂ、Ｒｓ＋ｂと、レジスタＲｔ、Ｒｔ＋１、Ｒｔ＋２、Ｒｔ＋３のそれぞれの積を、レジスタＲｄ、Ｒｄ＋１、Ｒｄ＋２、Ｒｄ＋３に格納する。ＭＡＣ４Ｂ命令は、レジスタＲｓ＋Ｒｂ、Ｒｓ＋ｂ、Ｒｓ＋ｂ、Ｒｓ＋ｂと、レジスタＲｔ、Ｒｔ＋１、Ｒｔ＋２、Ｒｔ＋３のそれぞれの積を、レジスタＲｄ、Ｒｄ＋１、Ｒｄ＋２、Ｒｄ＋３に足し込む。
【００４９】
以上のプロセッサを利用して、８＊８の離散コサイン変換、逆離散コサイン変換を行なう手順を以下に示す。
【００５０】
まず、行列データと係数データをレジスタにロードする。このロード作業は、Ｂブロックの離散コサイン変換処理の最初にただ１回だけ行えばよい。
【００５１】
＃行列、係数ロード
```
ＬＤ４　　ＣＯＥＦＦ，０，１，ＶＲ１２
ＬＤ４　　ＭＡＴＲＩＸ，０，４，ＶＲ１６
ＬＤ４　　ＭＡＴＲＩＸ，１，４，ＶＲ２０
ＬＤ４　　ＭＡＴＲＩＸ，２，４，ＶＲ２４
ＬＤ４　　ＭＡＴＲＩＸ，３，４，ＶＲ２８
```
次に８点１次元離散コサイン変換を行う命令列を示す。ＯＦＦはＩＭＧからのオフセットを示し第１回目の処理では０とする。
【００５２】
＃８点離散コサイン変換（横）
```
ＬＤ４　　ＩＭＧ，０＋ＯＦＦ，１，ＶＲ８
ＬＤ４　　ＩＭＧ，７＋ＯＦＦ，－１，ＶＲ４
ＡＤＤ４　ＶＲ８，ＶＲ４，ＶＲ０
ＳＵＢ４　ＶＲ８，ＶＲ４，ＶＲ４
ＭＵＬ４　ＶＲ４，ＶＲ１２，ＶＲ４
ＭＵＬ４Ｂ　ＶＲ０，ＶＲ１６，ＶＲ８，０
ＭＡＣ４Ｂ　ＶＲ０，ＶＲ２０，ＶＲ８，１
ＭＡＣ４Ｂ　ＶＲ０，ＶＲ２４，ＶＲ８，２
ＭＡＣ４Ｂ　ＶＲ０，ＶＲ２８，ＶＲ８，３
ＭＵＬ４Ｂ　ＶＲ４，ＶＲ１６，ＶＲ０，０
ＭＡＣ４Ｂ　ＶＲ４，ＶＲ２０，ＶＲ０，１
ＭＡＣ４Ｂ　ＶＲ４，ＶＲ２４，ＶＲ０，２
ＭＡＣ４Ｂ　ＶＲ４，ＶＲ２８，ＶＲ０，３
ＡＤＤ　　Ｒ１，Ｒ９，Ｒ１
ＡＤＤ　　Ｒ２，Ｒ１０，Ｒ２
ＡＤＤ　　Ｒ３，Ｒ１１，Ｒ３
ＳＴ４　　ＩＭＧ，０＋ＯＦＦ，２，ＶＲ８
```

ＳＴ４　　ＩＭＧ，１＋ＯＦＦ，２，ＶＲ０
ＯＦＦを８ずつ増加させながらこの１８命令で構成される変換処理を８回行なうことで、８＊８画素に対して横方向の１次元離散コサイン変換が完了する。その後、縦方向の変換を行うために、ＯＦＦを０、１、．．．、７と変化させながら以下の命令列を８回行なう。

【００５３】

＃８点離散コサイン変換（縦）

```
ＬＤ４　　ＩＭＧ，０＋ＯＦＦ，８，ＶＲ８
ＬＤ４　　ＩＭＧ，５６＋ＯＦＦ，－８，ＶＲ４
ＡＤＤ４　ＶＲ８，ＶＲ４，ＶＲ０
ＳＵＢ４　ＶＲ８，ＶＲ４，ＶＲ４
ＭＵＬ４　ＶＲ４，ＶＲ１２，ＶＲ４
ＭＵＬ４Ｂ ＶＲ０，ＶＲ１６，ＶＲ８，０
ＭＡＣ４Ｂ ＶＲ０，ＶＲ２０，ＶＲ８，１
ＭＡＣ４Ｂ ＶＲ０，ＶＲ２４，ＶＲ８，２
ＭＡＣ４Ｂ ＶＲ０，ＶＲ２８，ＶＲ８，３
ＭＵＬ４Ｂ ＶＲ４，ＶＲ１６，ＶＲ０，０
ＭＡＣ４Ｂ ＶＲ４，ＶＲ２０，ＶＲ０，１
ＭＡＣ４Ｂ ＶＲ４，ＶＲ２４，ＶＲ０，２
ＭＡＣ４Ｂ ＶＲ４，ＶＲ２８，ＶＲ０，３
ＡＤＤ　　Ｒ１，Ｒ９，Ｒ１
ＡＤＤ　　Ｒ２，Ｒ１０，Ｒ２
ＡＤＤ　　Ｒ３，Ｒ１１，Ｒ３
ＳＴ４　　ＩＭＧ，０＋ＯＦＦ，１６，ＶＲ８
ＳＴ４　　ＩＭＧ，５６＋ＯＦＦ，－１６，ＶＲ０
```

以上の操作により、８＊８の２次元離散コサイン変換を完了する。変換対象となるブロック数Ｂは十分大きいとすると、行列、係数ロードに必要な５命令を無視することができるため、１ブロックあたり２８８命令で２次元離散コサイン変換を処理できる。

【００５４】

（６）実施例４

本実施例は、本発明の処理手順をＣ言語で記述して利用する。係数、データは実施例１で示した図１５のように主記憶上に配置されているとする。

【００５５】

まず、図１８、図１９に示すように離散コサイン変換、逆離散コサイン変換に用いるデータタイプをＤＣＴＴＹＰＥとして定義し、同タイプの４＊４の２次元大域配列定数としてＭ［４］［４］、Ｃ４［４］を宣言する。Ｍには４＊４行列の値が、Ｃ４には４つの係数が設定されている。これは実施例１のレジスタＸＲ０～ＸＲ１５、Ｒ１２～Ｒ１５に相当する。次に、下位関数としてｌｄ４（）、ｓｔ４（）、ａｄｄ４（）、ｓｕｂ４（）、ｍｕｌ４（）、ｔｒｖ（）、を定義する。ｌｄ４（）はポインタａｄｒの示すメモリアドレスから４つのＤＣＴＴＹＰＥの値を取りだし、長さ４の１次元配列ＶＲに代入する。逆に、ｓｔ４はＶＲの値をポインタａｄｒの指し示すアドレスに書き込む。ａｄｄ４（）、ｓｕｂ４（）、ｍｕｌ４（）は、長さ４の１次元配列ＶＲ１、ＶＲ２の各要素について、それぞれ加算、減算、乗算を行ない、その結果をＶＲ３に代入する。ｔｒｖ（）は、４＊４の２次元配列Ｍで示される行列と、長さ４の１次元配列ＶＲ１の積を計算し、ＶＲ２に結果を代入する。これらの関数は、実施例１のＬＤ４、ＳＴ４、ＡＤＤ４、ＳＵＢ４、ＭＵＬ４、ＴＲＶに相当する処理を行なう。

【００５６】

以上の変数、下位間数をもとに、８点の離散コサイン変換、８＊８の２次元離散コサイン変換行なう関数は図２０のｄｃｔ８（）、ｄｃｔ８＿８（）ように書くことができる。このプログラムは使用するプロセッサごとに用意されたＣコンパイラが最適化を行なうため

、その実行性能はコンパイラ性能にも依存しているといえる。しかし、本発明のアルゴリズムは、８＊８の離散コサイン変換に４＊４の係数行列しか利用しないため、図７に示す８＊８行列演算をそのまま実装した、６４の係数を利用するプログラムと比べて、主記憶とプロセッサとのロード／ストアの回数を減らすことができ、コンパイラがより高速な機械語命令列を出力することが期待できる。実施例１、２のように、行列演算命令や、ＳＩＭＤ命令を搭載したプロセッサの場合には、コンパイラのビルトイン機能を用いて機械語で記述した場合により近い結果を得らる。また下位関数のみ機械語記述をすることも可能である。

【００５７】

（７）実施例５

本実施例は、実施例４の８点離散コサイン変換を利用して、１６点の離散コサイン変換を行う事例を示す。本発明の分割手法を用いて、１６点の変換を８点に分割し、８点の離散コサイン変換を実施例４の関数で処理することで、４＊４の係数行列を固定したまま１６点の離散コサイン変換を行う。

【００５８】

先ほど導いた式（９）、式（１０）、式（１１）、にＮ＝１６を代入してまとめると図２１に示すように、２つの８点離散コサイン変換に分割できることが分かる。これは、１６点離散コサイン変換がある前処理と後処理を追加することによって、２つの８点離散コサイン変換に分割できることを示すものである。

【００５９】

この特徴を利用して、１６点離散コサイン変換を図１８、図１９に示す関数と、図２０に示すｄｃｔ８（）関数を利用して記述すると図２２、図２３、図２４のようになる。

【００６０】

同様にして、３２点、６４点、．．．、２＾ｎ点の離散コサイン変換も４＊４係数行列を用いて求めることができる。

【００６１】

（８）実施例６

本実施例の変換装置は図１の変換装置のプログラム記憶装置１０９をプロセッサ１０２と同一チップに集積する点で実施例１と異なる。実行可能な命令、処理内容は全て実施例１と同一である。

【００６２】

（９）実施例７

本実施例の変換装置は図１の変換装置のデータ記憶装置１１０をプロセッサ１０２と同一チップに集積する点で実施例１と異なる。実行可能な命令、処理内容は全て実施例１と同一である。

【００６３】

（１０）実施例８

本実施例の変換装置の構成を図２５に示す。変換装置２５０１はプロセッサ２５０２によって構成される。プロセッサ２５０２はアドレス生成器１０６、レジスタファイル１０７、演算器１０８に加えてプログラム記憶装置１０９およびデータ記憶装置１１０が内蔵されている点で実施例１と異なる。実行可能な命令、処理内容は全て実施例１と同一である。

【００６４】

【発明の効果】

本発明は、Ｎ点の離散コサイン変換、逆離散コサイン変換を、ただ１つの固定された（Ｎ／２＾ｋ）＊（Ｎ／２＾ｋ）の係数行列を利用して計算することを可能とする。これにより、従来は行列演算ごとに必要であった係数行列の入れ替え操作を不要とし、そのオーバーヘッドを取り除くことができるため、演算器の利用効率が向上する。

【００６５】

また、演算の規模や係数の格納に必要なレジスタ数も減少し、回路面積を縮小することが

できる。これは、汎用のプロセッサに行列演算器やそれに類する演算装置を追加し、回路面積の増加を抑えながら高速な画像や音声の圧縮・伸長処理装置を構成することを可能とする。

【図面の簡単な説明】

【図１】本発明を適用したデータ変換装置。

【図２】本発明を適用した８点離散コサイン変換の分割（バタフライ図）。

【図３】本発明を適用した８点離散コサイン変換の分割（行列表現）。

【図４】従来技術の８点離散コサイン変換（行列表現）。

【図５】離散コサイン変換の定義式。

【図６】離散コサイン変換、逆離散コサイン変換で使用する係数ｃ（ｋ，Ｎ）の性質。

【図７】８点１次元離散コサイン変換の定義式に準じた行列表現。

【図８】本発明による離散コサイン変換の式変形。

【図９】本発明による８点離散コサイン変換の式変形（Ｘ［Ｎ－１］）。

【図１０】実施例１のレジスタの構成。

【図１１】実施例１におけるＴＲＶ命令の処理内容。

【図１２】実施例１におけるＡＤＤ４命令の処理内容。

【図１３】実施例１におけるＳＵＢ４命令の処理内容。

【図１４】実施例１におけるＭＵＬ４命令の処理内容。

【図１５】８＊８点２次元離散コサイン変換で使用するデータの主記憶配置。

【図１６】実施例２のレジスタの構成。

【図１７】実施例２におけるＩＰＲ命令の処理内容。

【図１８】実施例４の４＊４行列を利用した８点離散コサイン変換のＣ言語記述（下位関数１）。

【図１９】実施例４の４＊４行列を利用した８点離散コサイン変換のＣ言語記述（下位関数２）。

【図２０】実施例４の８点１次元離散コサイン変換、８＊８点２次元離散コサイン変換のＣ言語記述（上位関数）。

【図２１】本発明による１６点離散コサイン変換の分割（バタフライ図）。

【図２２】実施例５の４＊４行列を利用した１６点離散コサイン変換のＣ言語記述（下位関数）。

【図２３】実施例５の１６点１次元離散コサイン変換のＣ言語記述（上位関数）。

【図２４】実施例５の１６＊１６点２次元離散コサイン変換のＣ言語記述（上位関数）。

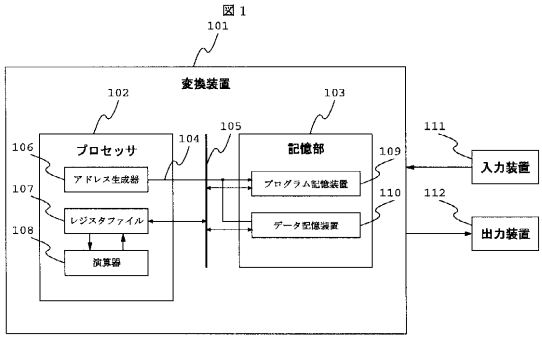
【図２５】実施例８の装置の構成。

【符号の説明】

１０１：変換装置
１０２：プロセッサ
１０３：記憶部
１０４：アドレスバス
１０５：データバス
１０６：アドレス生成器
１０７：レジスタファイル
１０８：演算器
１０９：プログラム記憶装置
１１０：データ記憶装置
１１１：入力装置
１１２：出力装置
２５０１：変換装置
２５０２：プロセッサ。

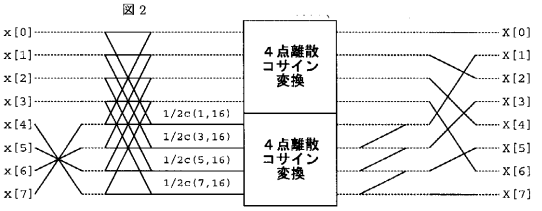
【図１】

図１



【図２】

図２



【図３】

図３

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \\ X[4] \\ X[5] \\ X[6] \\ X[7] \end{bmatrix} = \begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 0&0&0&0&1&1&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&1&1&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&1&1 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&1 \end{bmatrix} \cdot \begin{bmatrix} c(0,8) & c(0,8) & c(0,8) & c(0,8) & 0 & 0 & 0 & 0 \\ c(1,8) & c(3,8) & c(5,8) & c(7,8) & 0 & 0 & 0 & 0 \\ c(2,8) & c(6,8) & c(10,8) & c(14,8) & 0 & 0 & 0 & 0 \\ c(3,8) & c(9,8) & c(15,8) & c(21,8) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c(0,8) & c(0,8) & c(0,8) & c(0,8) \\ 0 & 0 & 0 & 0 & c(1,8) & c(3,8) & c(5,8) & c(7,8) \\ 0 & 0 & 0 & 0 & c(2,8) & c(6,8) & c(10,8) & c(14,8) \\ 0 & 0 & 0 & 0 & c(3,8) & c(9,8) & c(15,8) & c(21,8) \end{bmatrix} \cdot \begin{bmatrix} x[0]+x[7] \\ x[1]+x[6] \\ x[2]+x[5] \\ x[3]+x[4] \\ (x[0]-x[7])/2c(1,16) \\ (x[1]-x[6])/2c(3,16) \\ (x[2]-x[5])/2c(5,16) \\ (x[3]-x[4])/2c(7,16) \end{bmatrix} \tag{1}$$

【図４】

図４

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \\ X[4] \\ X[5] \\ X[6] \\ X[7] \end{bmatrix} = \begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 0&0&0&0&1&1&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&1&1&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&1&1 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&1 \end{bmatrix} \cdot \begin{bmatrix} c(4) & c(4) & c(4) & c(4) & 0 & 0 & 0 & 0 \\ c(2) & c(6) & c(-6) & c(-2) & 0 & 0 & 0 & 0 \\ c(4) & c(-4) & c(-4) & c(4) & 0 & 0 & 0 & 0 \\ c(6) & c(-2) & c(2) & c(-6) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c(1) & c(3) & c(5) & c(7) \\ 0 & 0 & 0 & 0 & c(3) & c(-7) & c(-1) & c(-5) \\ 0 & 0 & 0 & 0 & c(5) & c(-1) & c(7) & c(3) \\ 0 & 0 & 0 & 0 & c(7) & c(-5) & c(3) & c(-1) \end{bmatrix} \cdot \begin{bmatrix} 1&0&0&0&0&0&0&1 \\ 0&1&0&0&0&0&1&0 \\ 0&0&1&0&0&1&0&0 \\ 0&0&0&1&1&0&0&0 \\ 0&0&0&1&-1&0&0&0 \\ 0&0&1&0&0&-1&0&0 \\ 0&1&0&0&0&0&-1&0 \\ 1&0&0&0&0&0&0&-1 \end{bmatrix} \cdot \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ x[6] \\ x[7] \end{bmatrix}$$

$c(n,16)$ を $c(n)$ と省略

【図５】

図５

$$X[k] = A\sum_{n=0}^{N-1} x[n]c((2n+1)k, 2N) \tag{2}$$

$$A = \begin{cases} \frac{\sqrt{2}}{N} & k=0 \text{ のとき} \\ \frac{2}{N} & \text{others} \end{cases} \tag{3}$$

$$A = \mathbf{1} \tag{3'}$$

$$c(n,N) = \cos\left\{\frac{n\pi}{N}\right\} \tag{4}$$

【図６】

図６

$$c(n+N,N) = -c(n,N) \tag{5}$$

$$c(kn,kN) = c(n,N) \tag{6}$$

$$2c(n,N)c(m,N) = c(n+m,N) + c(n-m,N) \tag{7}$$

$$c\left(\frac{N}{2}k, N\right) = 0 \tag{8}$$

【図７】

図７

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \\ X[4] \\ X[5] \\ X[6] \\ X[7] \end{bmatrix} = \begin{bmatrix} c(0) & c(0) & c(0) & c(0) & c(0) & c(0) & c(0) & c(0) \\ c(1) & c(3) & c(5) & c(7) & c(9) & c(11) & c(13) & c(15) \\ c(2) & c(6) & c(10) & c(14) & c(18) & c(22) & c(26) & c(30) \\ c(3) & c(9) & c(15) & c(21) & c(27) & c(1) & c(7) & c(13) \\ c(4) & c(12) & c(20) & c(28) & c(4) & c(12) & c(20) & c(28) \\ c(5) & c(15) & c(25) & c(3) & c(13) & c(23) & c(1) & c(11) \\ c(6) & c(18) & c(30) & c(10) & c(22) & c(2) & c(14) & c(26) \\ c(7) & c(21) & c(3) & c(17) & c(31) & c(13) & c(27) & c(9) \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ x[6] \\ x[7] \end{bmatrix}$$

$c(n,16)$ を $c(n)$ と省略

【図８】

図８

$$\begin{aligned} X[2K] &= x[0]c(2K,2N) + x[1]c(6K,2N) + \dots \\ &\quad + x[N-1]c(\{2(N-1)+1\}2K,2N) \\ &= (x[0]+x[N-1])c(2K,2N) \\ &\quad + (x[1]+x[N-2])c(6K,2N) + \dots \\ &\quad + (x[N/2-1]+x[N/2])c(\{2(N/2-1)+1\}2K,2N) \\ &= (x[0]+x[N-1])c(K,N) \\ &\quad + (x[1]+x[N-2])c(3K,N) + \dots \\ &\quad + (x[N/2-1]+x[N/2])c((N-1)K,N) \end{aligned} \tag{9}$$

$$\begin{aligned} X[2K+1] &= x[0]c(2K+1,2N) + x[1]c(6K+3,2N) + \dots \\ &= (x[0]-x[N-1])c(2K+1,2N) \\ &\quad + x[N-1]c(\{2(N-1)+1\}(2K+1),2N) \\ &\quad + (x[1]-x[N-2])c(6K+3,2N) + \dots \\ &\quad + (x[N/2-1]-x[N/2])c((N-1)(2K+1),2N) \\ &= \frac{x[0]-x[N-1]}{2c(1,2N)}2c(1,2N)c(2K+1,2N) \\ &\quad + \frac{x[1]-x[N-2]}{2c(3,2N)}2c(3,2N)c(6K+3,2N) + \dots \\ &\quad + \frac{x[N/2-1]+x[N/2]}{2c(N-1,2N)}2c(N-1,2N)c((N-1)(2K+1),2N) \\ &= \frac{x[0]-x[N-1]}{2c(1,2N)}(c(K,N)+c(K+1,N)) \\ &\quad + \frac{x[1]-x[N-2]}{2c(3,2N)}(c(3K,N)+c(3(K+1),N)) + \dots \\ &\quad + \frac{x[N/2-1]-x[N/2]}{2c(N-1,2N)}(c((N-1)K,N)+c((N-1)(K+1),N)) \end{aligned} \tag{10}$$

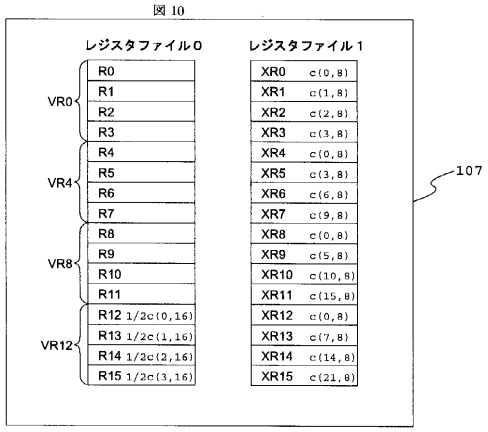
【図９】

図 9

$$
\begin{aligned}
X[N-1] &= \frac{x[0]-x[N-1]}{2c(1,2N)}(c(N/2\quad 1,N)\ |\ c(N/2,N)) \\
&\quad + \frac{x[1]-x[N-2]}{2c(3,2N)}(c(3(N/2-1),N)+c(3(N/2),N))+\ldots \\
&\quad + \frac{x[N/2-1]-x[N/2]}{2c(N-1,2N)}(c((N-1)(N/2-1),N)+c((N-1)(N/2),N)) \\
&= \frac{x[0]-x[N-1]}{2c(1,2N)}c(N/2-1,N) \\
&\quad + \frac{x[1]-x[N-2]}{2c(3,2N)}c(3(N/2-1),N)+\ldots \\
&\quad + \frac{x[N/2-1]-x[N/2]}{2c(N-1,2N)}c((N-1)(N/2-1),N) \qquad (11)
\end{aligned}
$$

【図１０】

図 10



【図１１】

図 11

$$
\begin{bmatrix} Rd \\ Rd+1 \\ Rd+2 \\ Rd+3 \end{bmatrix} = \begin{bmatrix} XR0 & XR4 & XR8 & XR12 \\ XR1 & XR5 & XR9 & XR13 \\ XR2 & XR6 & XR10 & XR14 \\ XR3 & XR7 & XR11 & XR15 \end{bmatrix} \begin{bmatrix} Rs \\ Rs+1 \\ Rs+2 \\ Rs+3 \end{bmatrix}
$$

【図１２】

図 12

$$
\begin{bmatrix} Rd \\ Rd+1 \\ Rd+2 \\ Rd+3 \end{bmatrix} = \begin{bmatrix} Rs \\ Rs+1 \\ Rs+2 \\ Rs+3 \end{bmatrix} + \begin{bmatrix} Rt \\ Rt+1 \\ Rt+2 \\ Rt\ |\ 3 \end{bmatrix}
$$

【図１３】

図 13

$$
\begin{bmatrix} Rd \\ Rd+1 \\ Rd+2 \\ Rd+3 \end{bmatrix} = \begin{bmatrix} Rs \\ Rs+1 \\ Rs+2 \\ Rs+3 \end{bmatrix} - \begin{bmatrix} Rt \\ Rt+1 \\ Rt+2 \\ Rt+3 \end{bmatrix}
$$

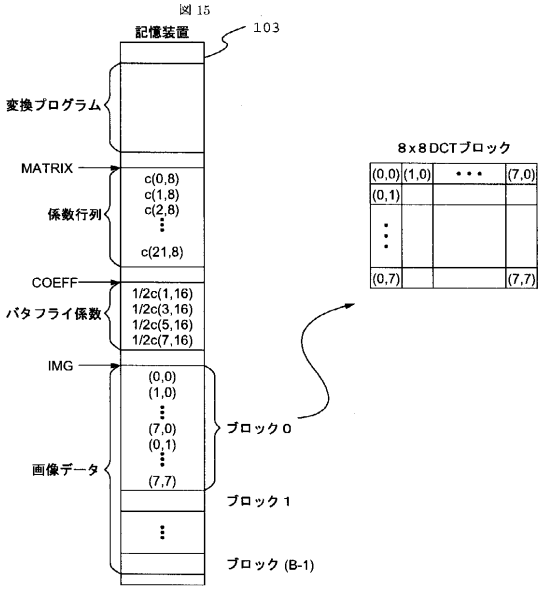
【図１４】

図 14

$$
\begin{bmatrix} Rd \\ Rd+1 \\ Rd+2 \\ Rd+3 \end{bmatrix} = \begin{bmatrix} Rs \cdot Rt \\ Rs+1 \cdot Rt+1 \\ Rs+1 \cdot Rt+2 \\ Rs+3 \cdot Rt+3 \end{bmatrix}
$$

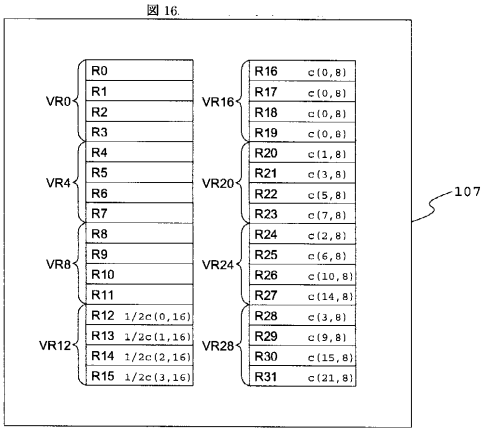
【図１５】

図 15



【図１６】

図 16



【図１７】

図 17

$$Rd = (Rs, Rs+1, Rs+2, Rs+3) \cdot (Rt, Rt+1, Rt+2, Rt+3)$$

【図１８】

図 18

```
DCTTYPE M[4][4]={ {c(0,8), c(0,8), c(0,8), c(0,8)},
                  {c(1,8), c(3,8), c(5,8), c(7,8)},
                  {c(2,8), c(6,8), c(10,8), c(14,8)},
                  {c(3,8), c(9,8), c(15,8), c(21,8)} };

DCTTYPE C4[4]={1/2c(1,16), 1/2c(3,16), 1/2c(5,16), 1/2c(7,16) };

void ld4( DCTTYPE *adr, int step, DCTTYPE *R ) {
  R[0]=*adr;adr+=step;
  R[1]=*adr;adr+=step;
  R[2]=*adr;adr+=step;
  R[3]=*adr;
}

void st4( DCTTYPE *adr, int step, DCTTYPE *R ) {
  *adr=R[0];adr+=step;
  *adr=R[1];adr+=step;
  *adr=R[2];adr+=step;
  *adr=R[3];
}
```

【図１９】

図 19

```
void add4( DCTTYPE *R1, DCTTYPE *R2, DCTTYPE *R3 ) {
  R3[0]=R1[0]+R2[0];
  R3[1]=R1[1]+R2[1];
  R3[2]=R1[2]+R2[2];
  R3[3]=R1[3]+R2[3];
}

void sub4( DCTTYPE *R1, DCTTYPE *R2, DCTTYPE *R3 ) {
  R3[0]=R1[0]-R2[0];
  R3[1]=R1[1]-R2[1];
  R3[2]=R1[2]-R2[2];
  R3[3]=R1[3]-R2[3];
}

void mul4( DCTTYPE *R1, DCTTYPE *R2, DCTTYPE *R3 ) {
  R3[0]=R1[0]*R2[0];
  R3[1]=R1[1]*R2[1];
  R3[2]=R1[2]*R2[2];
  R3[3]=R1[3]*R2[3];
}

void trv( DCTTYPE *R1, DCTTYPE *R2 ) {
  R2[0]=M[0][0]*R1[0]+M[0][1]*R1[1]+M[0][2]*R1[2]+M[0][3]*R1[3];
  R2[1]=M[1][0]*R1[0]+M[1][1]*R1[1]+M[1][2]*R1[2]+M[1][3]*R1[3];
  R2[2]=M[2][0]*R1[0]+M[2][1]*R1[1]+M[2][2]*R1[2]+M[2][3]*R1[3];
  R2[3]=M[3][0]*R1[0]+M[3][1]*R1[1]+M[3][2]*R1[2]+M[3][3]*R1[3];
}
```

【図２０】

図 20

```
void dct8( DCTTYPE *IMG, int skip ) {
  DCTTYPE VR1[4], VR2[4], VR3[4];

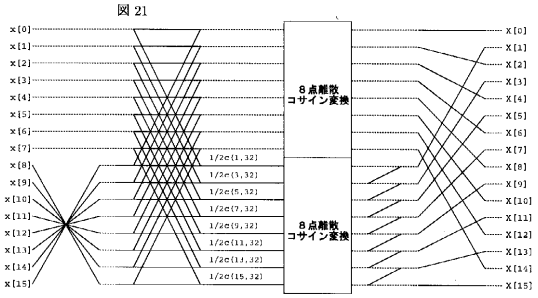
  ld4( IMG,         skip, VR3 );
  ld4( IMG+7*skip, -skip, VR2 );
  madd4( VR3, VR2, VR1 );
  msub4( VR3, VR2, VR2 );
  mmul4( VR2, C4, VR2 );
  trv( VR1, VR1 );
  trv( VR2, VR2 );
  VR2[0] += VR2[1];
  VR2[1] += VR2[2];
  VR2[2] += VR2[3];
  st4( IMG,      2*skip, VR1 );
  st4( IMG+skip, 2*skip, VR2 );
}

void dct8_8( DCTTYPE *IMG, int B ) {
  int i, j, off;

  for( i=0; i<B; i++ ) {
    for( j=0, off=0; j<8; j++, off+=8 )  /* Horizontal */
      dct8(IMG+off, 1);
    for( j=0, off=0; j<8; j++, off+=1 ) /* Vertical */
      dct8(IMG+off, 8);
    IMG+=64;
  }
}
```

【図２１】

図 21



【図２２】

図 22

```
DCTTYPE C8[8]={ 1/2c(1,32), 1/2c(3,32), 1/2c(5,32), 1/2c(7,32),
                1/2c(9,32), 1/2c(11,32), 1/2c(13,32), 1/2c(15,32) };

void ld8(DCTTYPE *adr, int step, DCTTYPE *R ) {
  ld4( adr,              skip, R );
  ld4( adr+skip*4        skip, R+4 );
}

void st8(DCTTYPE *adr, int step, DCTTYPE *R ) {
  st4( adr,              skip, R );
  st4( adr+skip*4        skip, R+4 );
}
```

【図２３】

図 23

```
void dct16( DCTTYPE *IMG, int skip ) {
  DCTTYPE VR1[8], VR2[8], VR3[8];

  ld8( IMG,              skip, VR3 );
  ld8( IMG+skip*15,     -skip, VR2 );
  madd8( VR3, VR2, VR1 );
  msub8( VR3, VR2, VR2 );
  mmul8( VR2, C8, VR2 );
  dct8( VR1, 1 );
  dct8( VR2, 1 );
  VR2[0] += VR2[1];
  VR2[1] += VR2[2];
  VR2[2] += VR2[3];
  VR2[3] += VR2[4];
  VR2[4] += VR2[5];
  VR2[5] += VR2[6];
  VR2[6] += VR2[7];
  st8( IMG,      2*skip, VR1 );
  st8( IMG+skip, 2*skip, VR2 );
}
```

【図２４】

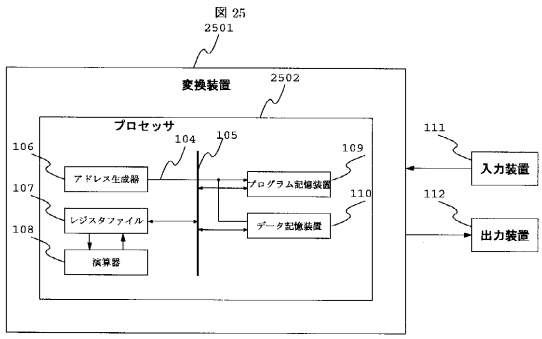
図 24

```
void dct16_16( DCTTYPE *IMG, int B ) {
  int i, j, off;

  for( i=0; i<B; i++ ) {
    for( j=0, off=0; j<16; j++, off+=16 )  /* Horizontal */
      dct16(IMG+off, 1);
    for( j=0, off=0; j<16; j++, off+=1 ) /* Vertical */
      dct16(IMG+off, 16);
    IMG+=256;
  }
}
```

【図２５】

図 25



フロントページの続き

審査官　鳥居　稔

(56)参考文献　国際公開第９９／００８２０４（WO，Ａ１）
特開２０００－１３７７０３（ＪＰ，Ａ）
K.R Rao，P. Yip，画像符号化技術　Discrete Cosine Transform Algorithms，Advantages，Applications，株式会社　オーム社，１９９２年　７月２０日，57-73

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 17/14
H03M 7/30
H04N 7/30