(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4158864号
(P4158864)
(45)発行日　平成20年10月1日(2008.10.1)　(24)登録日　平成20年7月25日(2008.7.25)

(51)Int.Cl.　　F I
G06F 17/16 (2006.01)　　G06F 17/16 F
G06F 15/167 (2006.01)　　G06F 15/167 610G

請求項の数 11　(全 13 頁)

(21)出願番号　特願平11-546748
(86)(22)出願日　平成11年2月22日(1999.2.22)
(65)公表番号　特表2002-510418(P2002-510418A)
(43)公表日　平成14年4月2日(2002.4.2)
(86)国際出願番号　PCT/IB1999/000315
(87)国際公開番号　WO1999/048025
(87)国際公開日　平成11年9月23日(1999.9.23)
審査請求日　平成18年1月18日(2006.1.18)
(31)優先権主張番号　98200867.4
(32)優先日　平成10年3月18日(1998.3.18)
(33)優先権主張国　欧州特許庁(EP)

(73)特許権者
コーニンクレッカ　フィリップス　エレクトロニクス　エヌ　ヴィ
オランダ国　５６２１　ベーアー　アインドーフェン　フルーネヴァウツウェッハ　1
(74)代理人
弁理士　津軽　進
(74)代理人
弁理士　沢田　雅男
(72)発明者　ファン　エイントホーベン　ヨゼフス　ティー　ジェー
オランダ国　５６５６　アーアー　アインドーフェン　プロフ　ホルストラーン　６

最終頁に続く

(54)【発明の名称】マトリックスのコサイン変換を計算するためのデータ処理装置およびその方法

(57)【特許請求の範囲】
【請求項１】
- オペランド内のそれぞれの位置に分割された複数のセグメントをそれぞれ有するオペランドを格納するオペランド格納回路と、
- 少なくとも１つのオペランドにおいて演算を実行する機能単位であって、前記オペランドの内容を、前記オペランドのそれぞれのセグメントに格納されている一連の数値として扱うことができ、複数のＡＬＵを有する機能単位と、
- 前記オペランド格納回路のそれぞれのソースオペランドをそれぞれ参照する１つまたは複数のオペランド参照を含む命令を実行する命令実行ユニットと、を有するデータ処理装置であって、前記命令が、前記機能単位に前記それぞれのソースオペランドの内容を受け取らせて、前記機能単位内の前記複数のＡＬＵに複数の演算を並列かつ互いに独立に実行させ、各演算が、１つまたは複数の前記それぞれのソースオペランドからのあらかじめ決められているセグメントの数値を算術的に組み合わせる、データ処理装置において、
- 前記演算が相異なる算術演算を含むことを特徴とするデータ処理装置。
【請求項２】
少なくとも前記演算の１つが、前記１つまたは複数のそれぞれのソースオペランド内の相異なる位置を有するセグメントの数値を算術的に組み合わせる、請求項１のデータ処理装置。
【請求項３】
前記命令実行ユニットが、前記オペランド格納回路内のそれぞれの他のソースオペランド

のセグメントをそれぞれ参照する複数のオペランド参照をさらに含む他の命令を実行するようにも構成され、当該他の命令が、前記命令実行ユニットに複数の他の演算を並列かつ互いに独立に実行させ、前記他の演算の各々が、参照される複数の前記他のソースオペランドからの、互いに対応する位置を有するセグメントの数値を算術的に組み合わせる、請求項２のデータ処理装置。

【請求項４】

少なくとも行と列を有するマトリックスの列変換と行変換の合成を計算するようにプログラムされていて、

- 各ソースオペランドが、同じ行及び相異なる列の数値を、当該列に対応する位置を有するセグメントに格納し、
- 前記列変換が、前記他の命令を使用して実行される１次元変換であり、前記複数の他のソースオペランドが、複数の前記各ソースオペランドであり、
- 前記行変換が、前記命令を使用して実行される１次元変換であり、前記１つまたは複数のソースオペランドが、１つまたは複数の前記各ソースオペランドである、

請求項３のデータ処理装置。

【請求項５】

前記行変換と前記列変換とが、前記行と前記列とを置き換えることにより同じ１次元変換に対応する、請求項４のデータ処理装置。

【請求項６】

前記命令によって行われる前記演算が、前記１つまたは複数のソースオペランド内の2つのセグメントの合計と差の計算を含む、請求項１のデータ処理装置。

【請求項７】

前記命令によって行われる前記演算により、前記１つまたは複数のソースオペランドの各セグメントに格納されている数値のベクトル変換の複数の成分係数が計算され、かつ前記データ処理装置が、前記命令によって参照される結果オペランドのそれぞれの位置のセグメントに前記成分係数を格納する、請求項１のデータ処理装置。

【請求項８】

前記計算が、結果的にIDCT又はDCTである、請求項７のデータ処理装置。

【請求項９】

前記複数のソースオペランドの前記セグメント内に格納されている前記数値が、変換される入力ベクトルを構成し、前記入力ベクトルの前記変換の前記成分係数が、複数の結果オペランドの前記セグメントに格納される、請求項７のデータ処理装置。

【請求項１０】

- オペランドを記憶するオペランド格納回路と、
- 複数のＡＬＵを有する機能単位と、
- 命令実行ユニットと、

を有するデータ処理装置が命令を実行する方法であって、

- 前記機能単位が、少なくとも１つのオペランドについて演算を実行し、前記オペランドの内容を、前記オペランドのそれぞれのセグメントに格納されている一連の数値として扱い、
- 前記命令実行ユニットが、前記オペランド格納回路のそれぞれのソースオペランドをそれぞれ参照する１つまたは複数のオペランド参照を含む命令を実行し、前記命令が、前記機能単位に前記それぞれのソースオペランドの内容を受け取らせて、前記機能単位内の前記ＡＬＵに複数の演算を並列かつ互いに独立に実行させ、各演算が、前記１つまたは複数のそれぞれのソースオペランドからのあらかじめ決められているセグメントを組み合わせる、方法において、
- 少なくとも前記演算の１つが、前記演算が相異なる算術演算を含むことを特徴とする方法。

【請求項１１】

請求項10の前記方法を実行するコンピュータプログラムを格納する、コンピュータによっ

て読み取ることのできるメディア。

【発明の詳細な説明】

技術分野

本発明は、請求項1の特徴記載部分に記載されているデータ処理装置に関する。

背景技術

このようなデータ処理装置は、PCT特許出願番号97/31308から公知である。このデータ処理装置は、SIMD(Single Instruction Multiple Data)命令などの並列命令の制御下での並列処理を可能とする。SIMD命令は、同じ演算を何度も並列に適用する。一般的にSIMD命令は、通常、レジスタアドレスにより2つのオペランドを定義する。これらの各オペランドの内容は、パック化データの複数のセグメントとして扱われる。例えば、64ビットレジスタの内容は、このレジスタ内のビット位置0-15、16-31、32-47、48-63における4つの16ビットの数値として扱うことができる。データ処理装置がSIMD命令に遭遇すると、同じ演算がオペランド内の数値のいくつかの相異なるペアに並列に適用される。例えば、第1オペランドレジスタ内のビット位置0-15の内容が、第2オペランドレジスタ内のビット位置0-15の内容に加えられ、第1オペランドレジスタ内のビット位置16-31の内容が、第2オペランドレジスタ内のビット位置16-31の内容に加えられる。

SIMD命令を使用すると、1つの機能を行うために実行しなくてはならない命令の数を減らすことができる。例えば、ピクセル値のブロックの個々の列の離散コサイン変換(IDCT)を実行するという機能を考察する。ブロックの相異なる行のピクセル値は、それぞれ相異なるオペランドに格納される。各オペランドにおいて、ピクセル値は、セグメント内の位置のうち、ピクセル値の列によって決まる位置に格納される。例えば、第1レジスタの場合、第1行、第1列のピクセル値はビット位置0-15に格納され、第1行、第2列のピクセル値はビット位置16-32に格納される。第2レジスタには、第2行からのピクセル値が格納され、以下同じように各行のピクセル値がそれぞれの列によって決まる位置に格納される。この結果、IDCTを1列に適用する演算用にコーディングされた一連の命令を実行すると、算術演算がすべてSIMD命令を使用して実行される場合、IDCTが自動的に多数の列について並列に実行される。これによって実行する必要のある命令の数が少なくなる。

分離可能な2次元IDCTの場合には、1次元IDCTをブロックの個々の列と個々の行に適用する必要がある。この場合、列の変換と行の変換の間で行と列の役割を交換すれば、同じように命令の数を減らすことができる。行と列の役割は、ブロックの置き換えによって交換できる。この置き換えにより、1つの列の相異なるピクセル値が、1つの行の相異なるピクセル値に代えて同じレジスタに格納される。つまり置き換えにより、相異なるレジスタの(同じ列に)対応する位置の内容が、別のレジスタ内の相異なる位置に移動される。しかしながら置き換えの実行自体には、かなりの数の命令を加えることが必要となる。このため2次元変換では、1次元変換に必要な命令数の2倍以上の命令数が必要となる。

SIMDの利点に対するこの制約は、パック化データの中の互いに対応しない位置にあるデータを組み合わせる必要がある機能をプログラムしなくてはならないときには、一般にはさらに大きくなる。この場合、互いに独立した数値を含むパック化フォーマットとしてオペランドの内容を扱うSIMD並列命令を使用することはできず、またSIMD演算を使用できるようにデータを並べ替えるための追加演算が少なくとも必要となる。

発明の開示

本発明の目的は、実行する必要のある命令の数をさらに減らすことができる、於て書きに記載した処理装置を提供することである。

本発明のデータ処理装置は、請求項1の特徴記載部分を特徴とする。これにより、オペランド内の互いに同じでない位置のセグメントを組み合わせ、または相異なる演算を使用して、オペランドのセグメントの相異なる組み合わせをつくる、並列演算をプログラムすることができる。これは、同じ位置のセグメントのペアに毎回同じ演算を適用する先行技術のSIMD命令とは対照的である。本発明の命令の場合、例えば、オペランドレジスタのビット位置0-15に格納されている数値が、ビット位置16-31に格納されている数値に加算され、それと並列に、ビット位置32-47に格納されている数値が、ビット位置48-63に格納され

ている数値に加算される。
同一オペランドレジスタ内のセグメントを組み合わせる演算と、異なるオペランドレジスタ内のセグメントを組み合わせる演算の両方に対する命令を提供することができる。1つまたは複数の如何なるセグメントも、2つ以上の演算で使用することができる。並列に実行される演算は、すべて同じタイプの演算(例：すべて加算)でも良いし、相異なる演算(例：加算と減算)でも良い。
通常、SIMD命令以外には、セグメントを組み合わせるアプリケーション固有命令は、ごく限られた命令セットしか提供されないであろう。例えば、相異なる位置の特定のセグメント間での加算などの演算を行う命令が利用可能なとき、可能なセグメントペアすべての間にその演算をプログラムする命令セットを用意する必要はない。同様に、(少なくとも演算の1つが他の演算と異なる)いくつかのあるセグメントペアをそれ自身の演算に組み合わせる命令が利用可能なとき、それらのセグメントに適用される演算の可能な組み合わせすべてについて命令セットを用意する必要はない。如何なるアプリケーションに対しても、可能な演算すべてまたは可能な演算の組み合わせすべてのうちのごく一部、および／またはセグメントの可能な組み合わせすべてのうちのごく一部しか必要とされない。
ブロックについての分離可能な2次元変換に対しては、本発明は、ブロックの置き換えなしに必要な命令の数を減らすことを可能にする。各レジスタは、1つの行からの異なるピクセル値を含み、かつ別のレジスタが同じ列からのピクセル値を同じセグメントに格納していてよい。列の変換は、SIMD命令を使用して実行されるであろうが、行の変換は、相異なるセグメント内の同じ行からのピクセル値を組み合わせる並列演算によって実行される。
例えば、IDCT命令の中で参照されるオペランドレジスタの異なるセグメントに格納されている行のピクセル値から、その行全体のIDCTを計算するIDCT命令を提供することもできる。また、レジスタ内の異なるセグメントのペアの内容の合計と差を計算する演算も用いることができる。この演算は、IDCT変換とこれに類似する変換で一般に必要となるタイプの演算である。
【図面の簡単な説明】
上述した本発明の利点とその他の利点について、以下の図を用い、例示的に説明する。
第1図は、データ処理装置を示す。
第2図は、8点1次元IDCTの実行のデータフロー図の例を示す。
第3図は、本発明による命令のデータフロー図を示す。
第4a図と第4b図は、本発明による命令を実行するための機能単位を示す。
発明を実施するための最良の形態
第1図は、VLIW(Very Long Instruction Word)タイプのデータ処理装置を示す。本発明は、VLIWタイプの装置を用いて図示されているが、このタイプの装置に限定されるものではない。この装置は、命令発行ユニット10、多数の機能単位12a-c、レジスタファイル14を有する。命令発行ユニット10は、機能単位12a-cとレジスタファイル14に結合された命令出力端を有する。レジスタファイル14は、機能単位12a-cのオペランド入力／出力端に結合された読み取り／書き込みポートを有する。
1つの機能単位12aが、詳細に図示されている。この機能単位12aは、命令デコーダ120、多数のALU(算術論理演算ユニット)122a～122d、第1入力レジスタ124aおよび第2入力レジスタ124b、出力レジスタ126を有する。命令デコーダは、ALU122a～122dに結合されている。入力レジスタ124aおよび124bは、多数のセグメントに分割されている。第1入力レジスタ124aおよび第2入力レジスタ124bのセグメントは、ALU 122a-dに接続されている。
動作時には、命令発行ユニット10が、プログラムの中の連続する命令にアクセスし、それらの命令を機能単位12a-cに発行する。機能単位12a-cに発行される命令は、代表的には、1つの演算コード、2つのソースレジスタアドレス、1つの結果レジスタアドレスを有する(命令のこれらの要素は必ずしも同時に発行されなくてもよい)。演算コードは、機能単位12a-cが実行しなければならない1つまたは複数の演算を定義する。ソースレジスタアドレスは、その1つまたは複数の演算を実行する対象のオペランドが格納されるレジスタファ

イル14の中のレジスタを参照する。命令発行ユニット10は、これらのアドレスをレジスタファイル14に適用する。結果レジスタアドレスは、1つまたは複数の演算の結果が格納されるレジスタファイル14の中のレジスタを参照する。命令発行ユニット10は、結果レジスタアドレスをレジスタファイル14に適用する。

機能単位12a-cのほとんどは、各レジスタの内容を1つの数値として扱う。例えば、レジスタの長さが64ビットの場合、その内容は、別の64ビットの数値に加えたり、算術的あるいは論理的にシフト等ができる64ビットの数値として扱われる。しかしながら機能単位12a-cの少なくともいくつかは、レジスタの内容を、そのレジスタの各セグメントに格納されている一連の数値として扱うことが(または扱うことも)できる。次のような専用演算を、これらの数値に対し互いに無関係に並列に実行することができる。つまり、キャリービットによって1つのセグメントから別のセグメントに桁上げされず、シフトによって1つのセグメントから別のセグメントにビットが桁送りされず、クリッピングが各セグメントごとに独立して行われる等である。

機能単位12aは、各レジスタの内容を、それぞれ個別の数値が格納されている複数のセグメントとして扱う機能単位である。この目的のため、すべてのレジスタは、同じ方式で仮想的にセグメントに分割される。命令が実行されると、その命令内で参照されるソースレジスタの各セグメントの内容が、ALU 122a-dのそれぞれに適用される。

SIMD命令の場合、2つのソースオペランド内の同じ位置にあるセグメントの内容が、同じALU 122a-dに供給される。例えば、オペラントが64ビットであり、ビット位置0-15、16-31、32-47、48-63は、4つのセグメントS0、S1、S2、S3をそれぞれ構成するとする。このとき両方のオペランドのビット位置0-15の内容は最初のALU 122aに供給され、ビット位置16-31の内容は2番目のALU 122bに供給される（以下同様）。またSIMD命令の場合、命令デコーダ120は、ALU 122a-dのすべてに同じ制御コードを適用する。このためALU 122a-dは、すべて同じタイプの演算(例：加算)をそれぞれ別のセグメントに実行する。

SIMD命令は、例えば、数値B_{ij}(i=0..n、j=0..m)のブロックB(例：8×8ブロック(n=7、m=7))の多数の列の1次元変換の計算に適用できる。そのために、ブロックの同じ行の数値は、1つのレジスタの各セグメントにロードされる。例えば、数値$B_{0,0}$、$B_{0,1}$、$B_{0,2}$、$B_{0,3}$は、第1レジスタR1のセグメントS0、S1、S2、S3にそれぞれロードされ、数値$B_{0,4}$、$B_{0,5}$、$B_{0,6}$、$B_{0,7}$は、第2レジスタR2のセグメントS0、S1、S2、S3にそれぞれロードされ、数値$B_{1,0}$、$B_{1,1}$、$B_{1,2}$、$B_{1,3}$は、第3レジスタR3のセグメントS0、S1、S2、S3にそれぞれロードされ、数値$B_{1,4}$、$B_{1,5}$、$B_{1,6}$、$B_{1,7}$は、第4レジスタR4のセグメントS0、S1、S2、S3にそれぞれロードされる。

ここで、1列の変換を実行するプログラムがあり、このプログラムは、1列の数値B_{ij}(i=0..n)が格納されているレジスタに適用される加算、減算、乗算などの算術命令を含む命令で表現されていると仮定する。これらの算術命令すべてにSIMD命令が使われる場合、このプログラムは、多数の列j=0..3について並列に変換を自動的に計算する。従って、列数がN、各レジスタの各セグメントにP個の数値が格納されているブロックの場合、N個の列を変換するには、このプログラムをN/P回のみ実行すればよい。

分離可能な2次元変換の場合には、すべての列は上述のように変換できる。次いで、変換後のブロックの行すべてを変換する必要がある。このような2次元変換の例として、2次元IDCTがある。この場合、変換後のブロックA_{ij}は次のように表わされる。

$$A_{ij}=2/N\ \Sigma_u\Sigma_v C_u C_v B_{u,v}\cos((2i+1)u\pi/2N)\cos((2j+1)v\pi/2N)$$

ここで、u=0の場合C_u=1/sqrt(2)、それ以外ではC_u=1であり、かつ0からN-1までの整数について合計が実行される。この2次元変換は、最初に、

$$INT_{iv}=\Sigma_u C_u B_{u,v}\cos((2i+1)u\pi/2N)$$

という1次元変換によって中間ブロックINT_{iv}を得、次いでこの中間ブロックに次の1次元変換

$$A_{ij}=2/N\Sigma_v C_v INT_{i,v}\cos((2j+1)v\pi/2N)$$

を適用することにより計算できる。

従って、この2次元変換は、2つの1次元変換、つまりBをINTにする変換とINTをAにする変

換の合成として計算される(2つの変換の「合成」とは、1つの変換の適用結果に別の変換が適用されることを意味する)。IDCTの例では、どちらの1次元変換が最初に適用されるかは問題ではない。本例では、まずブロック$B_{u,v}$の最初の添字に沿って合計を計算し、次に2番目の添字vに沿って合計を計算しているが、この順序を逆にしても最終結果は同じである。

このような2段階の2次元変換は、SIMD命令を使うことにより計算速度を高めることができる。中間ブロックBの数値$B_{u,v}$が前述したように格納されるとき、つまり同じ行の数値$B_{u,v}$(v=0,1,2,3)がレジスタの各セグメントに格納されるときには、中間ブロック$INT_{i,v}$の計算は、多数の列を並列に（第1列の中のv=0の数値すべて、第2列の中のv=1の数値すべて、以下同様）変換することにより実行できる。

例えば、第1レジスタのセグメントに$INT_{i,v}$(i=0..3、v=0)がそれぞれ格納され、第2レジスタのセグメントに$INT_{i,v}$(i=4..7、v=0)がそれぞれ格納され、第3レジスタのセグメントに$INT_{i,v}$(i=0..3、v=1)がそれぞれ格納される（以下同様）ように1つの列のいくつかの数値が1つのレジスタに格納されるように中間ブロックの数値がレジスタに格納されている場合には、SIMD命令を使用して同様の並列処理を行うことが可能である。この場合、中間ブロックINTの多数の行は、SIMD命令を使用して並列に変換できる。

しかしながら、ブロックBからの中間ブロックINTの計算後には、数値は、レジスタにこのように格納されることはないであろう。つまり、1つの列のいくつかの数値$INT_{i,v}$(i=0..3 v=0)が1つのレジスタに格納されるのではなく、1つの行のいくつかの数値$INT_{i,v}$(i=0 v=0..3)が各レジスタに格納されるであろう。この理由は、中間ブロックの計算時には各列について個別に1次元変換を行う必要があるのに対し、最終ブロックAの計算時には各行について個別に1次元変換を行う必要があるためである。

両方のタイプの変換にSIMD命令を使えるようにするには、中間ブロックを置き換える、つまりレジスタ間で数値を再編成する必要がある。これは複雑な操作である。4セグメントレジスタ、8×8ブロックの例では、置き換えのために16個のレジスタと、2つの入力をもつ32個の演算が必要となる。

本発明の目的は、この置き換えを不要とすることにある。行の変換に対して、同じ行の異なる数値がレジスタに格納されている中間ブロックの数値の配置が、そのま維持され、かつこれらのレジスタに格納されている行における1次元変換を実行するために、これらのレジスタ内のこれらの数値を組み合わせる専用命令が使用される。

専用命令を使用することにより、置き換えを行わずに2次元の分離可能な変換を実行することできる。また3次元以上の変換も、1次元用のこの専用命令と、2次元以上用のSIMDタイプの演算との組み合わせにより、他の方策を必要とせずに使用することができる。

もっとも単純な実行例では、1行のIDCT全体を実行できる機能単位が少なくとも1つ設けられる。1つの列の4つの数値をそれぞれ格納するレジスタを使用する8点IDCTの場合、このような命令は、2つのオペランドレジスタと2つの結果レジスタを必要とする。

第2図は、8点1次元IDCTの実行のデータフロー図の例を示す。このデータフロー図は、Proceedings International Conference on Acoustics, Speech and Signal Processing 1989(IC-IASSP‘89)の第988～991頁に刊行された、C. Loeffler、A. Ligtenberg、G. Moschytzらによって発表された「Practical Fast 1-D DCT Algorithms with 11 multiplications」という題名の記事の記載に基づいている。左側のノード30a-hは、変換する必要のある行における位置v=0..7の添字vの値により数値を表わしている。右側ノード32a-hは、変換後の行における位置j=0..7の添字jの値により変換後の数値を表している。ノード32a-hからの線は、数値から各演算までのデータフローと、これらの演算の結果から別の演算または変換後の数値までのデータフローを表す。演算は次のように表わされる。2本の実線が入力しているドットは合計を表す。1本の実線と1本の破線が入力しているドットは減算を表し、実線に沿って送られる数値から破線に沿って送られる数値が引かれる。2つの入力と2つの出力をもつボックスは回転と因数分解、つまり次の式に従う(X0,Y0)から(X1,Y1)への計算を表す。

$$X_1=I(X_0\cos\psi-Y_0\sin\psi)$$

$Y_1=I(X_0\sin\psi+Y_0\cos\psi)$
ボックス内には係数Iの値と角度∧の識別が記されていて、これらはあらかじめ決められた値である。ブロックは、4つの乗算、加算、減算を使用して実行できる(これに代えて、3つの乗算と3つの加算を使うこともできる)。
1つの実行例の場合、行IDCT命令によってそのオペランドのセグメントの内容をIDCT変換させる行IDCT命令を実行できる少なくとも1つの機能単位が設けられている。レジスタあたり4セグメントの8点IDCTの例では、行を変換するために2つのオペランドが必要となる。そしてこのような命令は、変換を表す数値がその変換における周波数位置(frequency position)に従って各セグメントに書き込まれる結果レジスタを2つ必要とする。
このような機能単位によるIDCTの実行は、個々の命令による実行よりもはるかに高速である。その理由は、少なくとも、オペランド内の異なる位置のセグメントに格納されている数値の組み合わせを、機能単位内の配線によって実現できることにある。この配線は、IDCTに固有な配線である。さらに、第3図のデータフロー図は、このような機能単位ではかなり多くの並列処理が可能であるため、多数の演算を並列に実行することにより実行速度をさらに高速にすることが可能であることを示している。
このように、2次元IDCT変換は、列については算術SIMD命令を使用して、多数の列に1次元IDCT変換を並列に適用し、かつ行についてはこれとは別の専用IDCT命令を使用して、機能的に同じIDCT変換を行に適用することにより実行することができる。
プロセッサアーキテクチャによっては、機能単位が、代表的には1つの演算コード、2つのソースレジスタ参照、1つの結果レジスタ参照を含む標準の命令フォーマットを使用する必要がある。この場合、各機能単位は、レジスタファイルの読み取りポートに接続される2つのポートと、レジスタファイルの書き込みポートに接続される1つのポートを有することが出来る。しかしながら、2つ以上のレジスタに格納されている数値を変換するIDCT命令の場合、変換後の数値を書き込むために2つ以上の結果レジスタが必要となる。1つの結果レジスタしか使用できないアーキテクチャの場合、この要件は、論理上隣接する結果レジスタに時間をずらして結果を順次書き込むなどのさまざまな方法で実現できる。これに代えて、機能単位に並列に発行される2つの命令の組み合わせを使用することもできる。通常、このような2つの命令は、2つの機能単位に対し並列に使用されるであろう。これに代えて、IDCTを実行する1つの機能単位をプログラムする2つの命令の組み合わせが使用される。この2つの命令の組み合わせを使用することにより、相異なる2つの結果レジスタを指定できる。並列に発行される各命令に対しレジスタファイルへの1つの書き込みポートを備えるプロセッサの場合、この方法によって、レジスタファイルへの1つの書き込みポートを両方の結果に対し確実に利用することが可能となる。
これに代えて、機能単位に対し2種類の命令、つまりレジスタ内の数値の半分を生成する命令と、残りの半分を生成する命令とを定義することもできる。
より一般的には、どの命令も結果レジスタの最大数(例：1つ)より多くの結果レジスタを必要としないように、IDCTの計算の部分ごとに専用の命令を用意してもよい。このような命令を選択するためには、IDCTデータフロー図を部分図に分け、1つの専用命令を各部分図に割り当てても良い。すべての部分図の出力数を一定数以下とすることにより、どの専用命令についても2つ以上の結果レジスタは不必要となるようにすることができる。
第3図は、破線のボックス39a-gによって示される部分図への分割の例を示す。各ボックスは、変換の計算速度を高めるために並列に実行される演算の組み合わせを提供する多数の専用命令のデータフローを定義する。各命令の結果に必要なセグメント数は4以下になっている。これらの命令は、各セグメント内の数値の場所がSIMD変換に必要な場所に対応するように、つまり第3図の左側のv=0..3によって示される第1レジスタの各セグメント内の数値と、u=4..7によって示される第2レジスタの各セグメント内の数値に対応するように定義される。
第１の例である、第1破線ボックス39aに対応する最初の命令INS1 R1,R2,R3は、オペランドとして2つのレジスタR1、R2を参照する。この命令により、機能単位は以下の演算を並列に実行する。

- 第1レジスタR1の第1セグメント内の数値(v=0)と第2レジスタR2の第1セグメント内の数値(v=4)とを加算する。その結果を結果レジスタR3の第1セグメントに格納する。
- 同じこれらの2つの数値の減算を行い、その結果を結果レジスタR3の第2セグメントに格納する。
- 第1レジスタR1の第3セグメント(v=2)内の数値と第2レジスタR2の第3セグメント内の数値を、係数sqrt(2)およびあらかじめ決められたサイン値とコサイン値の回転におけるX_0、Y_0として使用する。その結果であるX_1、Y_1を、結果レジスタの第3セグメントと第4セグメントに格納する。

第4b図は、INS1命令を実行するための機能単位40の例を示す。機能単位40は、第1レジスタR1と第2レジスタR2の内容をそれぞれ受け取る2つの入力セクション42、46と、この機能単位を作動させる命令デコーダ48、それにR1とR2の第1セグメントS0の合計、R1とR2の第1セグメントの差、R1とR2の第3セグメントS2の回転を計算する演算回路44a-cを有する。これらの計算の結果は、結果レジスタR3への書き込みのために、出力セクション49のセグメントS0～S3に結合される。

2番目の例である、第2破線ボックス39bに対応する2番目の命令INS2 R3,R4は、オペランドとして1つのレジスタR3を参照する。この命令により、機能単位は以下の演算を並列に実行する。

- オペランドレジスタR3の第1セグメントと第4セグメントに格納されている数値を加算し、その結果を結果レジスタR4の第1セグメントに格納する。
- オペランドレジスタR3の第2セグメントと第3セグメントに格納されている数値を加算し、その結果を結果レジスタR4の第2セグメントに格納する。
- オペランドレジスタR3の第2セグメント内の数値からオペランドレジスタR3の第3セグメント内の数値を減算し、その結果を結果レジスタR4の第3セグメントに格納する。
- オペランドレジスタR3の第1セグメント内の数値からオペランドレジスタR3の第4セグメント内の数値を減算し、その結果を結果レジスタR4の第4セグメントに格納する。

第4a図は、INS2命令を実行する機能単位20の例を示す。機能単位20は、オペランドレジスタR3の内容を受け取る入力セクション、加算と減算を計算する演算装置24a-bと25a-b、機能単位20を作動させる命令デコーダ28、出力セクション26を有する。加算と減算の結果は、結果レジスタR4への書き込みのために出力セクション26のセグメントS0～S3に結合される。

3番目の例である、第3破線ボックス39cに対応する3番目の命令INS3 R4,R9,R5は、オペランドとして2つのレジスタR4、R9を参照する。この命令により、機能単位は以下の演算を並列に実行する。

- 第1オペランドレジスタR4の第1セグメントとオペランドレジスタR9の第4セグメントに格納されている数値を加算し、その結果を結果レジスタR5の第1セグメントに格納する。
- 第1オペランドレジスタR4の第2セグメントと第2オペランドレジスタR9の第3セグメントに格納されている数値を加算し、その結果を結果レジスタR5の第2セグメントに格納する。
- 第1オペランドレジスタR4の第3セグメントと第2オペランドレジスタR9の第2セグメントに格納されている数値を加算し、その結果を結果レジスタR5の第3セグメントに格納する。
- 第1オペランドレジスタR4の第4セグメントと第2オペランドレジスタR9の第1セグメントに格納されている数値を加算し、その結果を結果レジスタR5の第4セグメントに格納する。

4番目の例である、破線ボックス39hに対応する4番目の命令INS4 R4,R9,R6は、オペランドとして2つのレジスタR4、R9を参照する。この命令により、機能単位は以下の演算を並列に実行する。

- オペランドレジスタR9の第4セグメントに格納されている数値から、第1オペランドレジスタR4の第1セグメントに格納されている数値を減算し、その結果を結果レジスタR6の第4セグメントに格納する。

- 第2オペランドレジスタR9の第3セグメントに格納されている数値から、第1オペランドレジスタR4の第2セグメントに格納されている数値を減算し、その結果を結果レジスタR6の第3セグメントに格納する。
- 第2オペランドレジスタR9の第2セグメントに格納されている数値から、第1オペランドレジスタR4の第3セグメントに格納されている数値を減算し、その結果を結果レジスタR6の第2セグメントに格納する。
- 第2オペランドレジスタR9の第1セグメントに格納されている数値から、第1オペランドレジスタR4の第4セグメントに格納されている数値を減算し、その結果を結果レジスタR6の第4セグメントに格納する。

5番目の例である、第4破線ボックス39dに対応する5番目の命令INS5 R1, R2, R7は、オペランドとして2つのレジスタR1、R2を参照する。この命令により、機能単位は以下の演算を並列に実行する。

- 第1ソースレジスタR1の第4セグメントと第2ソースレジスタR2の第2セグメントからの数値を、結果レジスタR7の第2セグメントと第3セグメントにそれぞれ格納する。
- 第2レジスタR2の第4セグメント(v=3)の中の数値と第1レジスタR1の第2セグメントの中の数値を、係数2およびあらかじめ決められたサイン値とコサイン値(45°に対応)の回転におけるX_0、Y_0として使用する。その結果のX_1、Y_1を結果レジスタの第1セグメントと第4セグメントに格納する(この回転は、45°のサインとコサインは互いに等しいので少ない回数の乗算で実行できる)。

6番目の例である、第6破線ボックス39eに対応する6番目の命令INS6 R7,R8は、オペランドとして1つのレジスタR7を参照する。この命令により、機能単位は以下の演算を並列に実行する。

- オペランドレジスタR7の第1セグメントと第3セグメントに格納されている数値を合計し、その結果を結果レジスタR8の第1セグメントに格納する。
- オペランドレジスタR7の第2セグメントと第4セグメントに格納されている数値を合計し、その結果を結果レジスタR8の第4セグメントに格納する。
- オペランドレジスタR7の第1セグメント内の数値からオペランドレジスタR7の第3セグメント内の数値を減算し、その結果を結果レジスタR8の第3セグメントに格納する。
- オペランドレジスタR7の第4セグメント内の数値からオペランドレジスタR7の第2セグメント内の数値を減算し、その結果を結果レジスタR8の第2セグメントに格納する。

7番目の例である、第7破線ボックス39fに対応する7番目の命令INS7 R8,R9は、オペランドとして1つのレジスタR8を参照する。この命令により、機能単位は以下の演算を並列に実行する。

- ソースレジスタR8の第1セグメントと第4セグメント内の数値を、係数sqrt(2)およびあらかじめ決められたサイン値とコサイン値の回転におけるX_0、Y_0として使用する。その結果のX_1、Y_1を結果レジスタR9の第1セグメント第4セグメントに格納する。
- ソースレジスタR8の第2セグメントと第3セグメント内の数値を、係数sqrt(2)およびあらかじめ決められたサイン値とコサイン値の回転におけるX_0、Y_0として使用する。その結果のX_1、Y_1を結果レジスタR9の第2セグメント第3セグメントに格納する。

これらの命令において、乗算時にいくつかの最下位ビットが破棄されるように、レジスタ内の数値をすべてビット数が同じ固定小数点数として表わしてもよい。ほとんどすべての固定小数点数は、+1～-1の範囲で定義することが出来る。しかしながら、回転／スケーリングの結果はこの例外であり、-2～2の範囲の固定小数点数が望ましい。この数値表現を使用しかつデータフロー図を前述したように命令に分割すると、丸めにより精度は、ほとんど失われないことが判明している。これらの命令における加算および／または乗算は、結果の大きさがレジスタに保持できる値範囲を超える場合には、命令の結果のクリッピングを提供することが望ましい。しかしながら、先に示したようにデータフロー図が命令に分割されているときは、通常、クリッピングは必要ないことが判明している。

これらの命令すべてがデータ処理装置によって提供される場合には、2つのレジスタR1、R2のセグメントに含まれる行の8点IDCTは、次のプログラムを使用してプログラムできる。

INS1 R1,R2,R3
INS2 R3,R4
INS5 R1,R2,R7
INS6 R7,R8
INS7 R8,R9
INS3 R4,R9,R5
INS4 R4,R9,R6
この結果として、IDCT変換の行を構成する数値は、レジスタR5、R6のセグメントに格納される。ブロック全体を変換するためには、必要となる他のレジスタを使用して、これらの命令を他の行についても繰り返す必要がある。また言うまでもないが、2つ以上の機能単位を有するVLIWプロセッサの場合、命令INS1-INS7すべてを1つの同じ機能単位に対する命令とすることが出来るが、これらの命令INS 1-INS7を複数の機能単位によって実行させることもできる。例えば、乗算を行う命令のための機能単位と、加算と減算のみを行う命令のための機能単位をそれぞれ用意することもできる。
命令への演算のグループ分けを再編成することも可能である。例えば、INS1 R1,R2,X; INS2 X,R4の順次実行とINSA R1,R2,R4の実行とが機能的に等価となるように、INS1とINS2の演算を1つの命令INSAに結合することができる。同様に、INS5 R1,R2,X; INS6 X,Y; INS7 Y,R9の順次実行とINSB R1,R2,R9の実行とが等価となるように、INS5、INS6、INS7を1つの命令に結合することができる。また演算の結果を逆の順序で結果レジスタのセグメントに格納するように命令INS7を修正することにより、INS3とINS4をそれぞれSIMD加算、SIMD減算に置き換えることができる。しかしながらこの場合、セグメント0-3の内容を互いに交換し、セグメント1-2の内容を互いに交換する追加の「逆順」命令が必要になる。この命令は、変換後の数値が正しい順序で得られるようにINS4のSIMDバージョンの結果に適用する必要がある。
ブロックを変換するために実行する必要のある命令の数は、命令INS1-INS7を受け取って演算を並列に実行し、命令内で参照される1つまたは複数のオペランド内の相異なるセグメントを組み合わせる1つまたは複数の機能単位を提供することにより減らすことができる。これにより変換に必要な時間(命令サイクルの数)が短縮される。このような機能単位によるIDCTの実行は、個々の命令による実行よりもはるかに高速である。その理由は、少なくとも、オペランド内の異なる位置のセグメントに格納されている数値の組み合わせを機能単位内の配線によって実現できることにある。この配線はIDCTに固有なものである。もちろん、追加命令INS1-INS7のうちの1つ、またはこれらの命令の任意の組み合わせが機能単位によって提供されている場合には、実行時間の短縮はすでに達成されている。これらの命令のうち提供されていない命令が1つ以上あるときは、それらの機能は従来の命令を使用して実行できる。
さらに、プログラムを格納するのに必要なメモリ空間も減少し、これは特に、変換が行われるプログラムで顕著である。当然ながらこの利点は、命令の中の演算が並列に実行されない場合にも得られる。どのような組み合わせの命令に対してもプログラム空間は減少するであろう。しかしながら、INS1-INS7の組み合わせは任意ではなく、それらは、IDCTの計算に必要なセグメントを組み合わせて処理速度を高めるための演算と、さらに並列に実行できる演算を組み合わせてIDCTの計算速度をさらに高めるための演算とを提供する特殊な特性を持つ。
先に示した例は、8点2次元IDCTを実行するのに4つのセグメントをもつレジスタ、例えば、4つの16ビットセグメントをもつ64ビットのレジスタを使用している。当然ながら、本発明は、これらの数値に限定されるものではない。これ以外のサイズのセグメント(例：8ビット、12ビット、32ビットのセグメント)および／またはこれ以外のビット数のレジスタ(例：128ビット)も使用できる。16ビットセグメントの128ビットレジスタを使用する場合、8つの数値を格納でき、例えば、8点IDCTおよび8ビットブロックの行全体を、1つのレジスタと1つの結果レジスタのみの1つの命令として実行できる。
より一般的には、レジスタ内の異なる位置のセグメントに格納されているオペランドを組

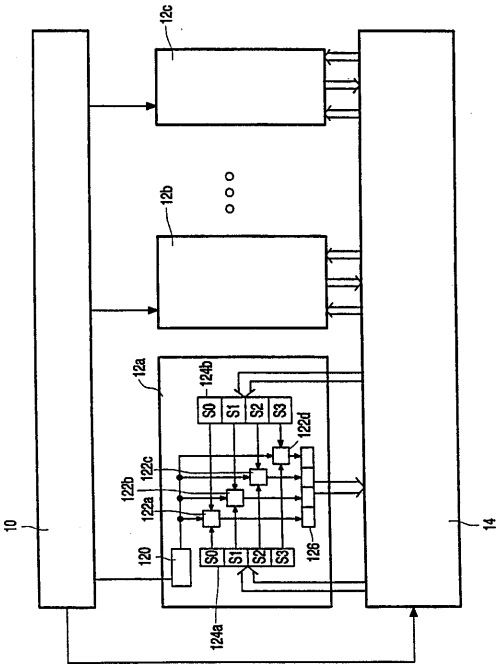
み合わせる演算を実行する(並列実行が望ましい)専用命令を実行できる機能単位を提供することにより、如何なる種類のプログラムの速度も高めることができる。前述した分離可能な変換は、その一例である。1つのプログラムが与えられたとき、そのプログラムのデータフローを分析し、同じオペランドまたは2つのオペランド内の異なるセグメントを組み合わせる演算の頻度の高い組み合わせを取り出すことにより、適切な専用命令を見出すことができる。適切な命令が見出された場合、命令デコーダ120とスイッチ回路125は、機能単位がその命令を扱えるように設計される。

これらの専用命令は、SIMD命令セットと組み合わせるのが望ましい。この場合、1つまたは複数の機能単位は、協働して、またはそれぞれ個別に、(オペランド内の対応する位置のセグメントのペアを組み合わせて)SIMDデータフローによる算術命令の完全なセットを提供する。さらに少なくとも1つの機能単位は、命令の1つまたは複数のオペランド内の異なる位置のセグメントを組み合わせる2～3個の選択された命令を実行できる（この場合の「異なる位置」とは、SIMD命令におけるオペレータランド内の位置と異なるという意味である）。

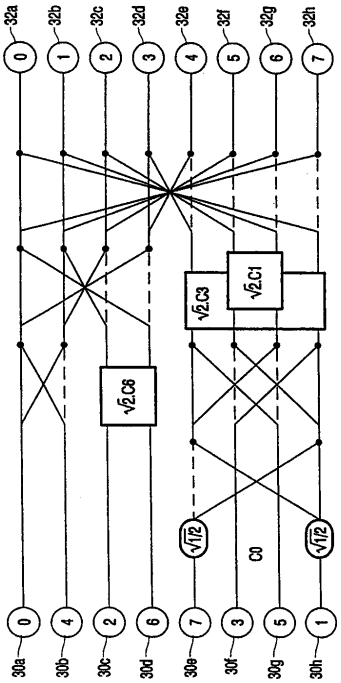
この方法は、IDCTのみならず、あらゆる種類の分離可能な変換に特に使用できる。例えば、2次元フーリエ変換やHadamard変換のほか、H1(x)H2(y)と書くことのできる(ガウスカーネルのような)2次元の分離可能なカーネルH(x,y)によるたたみこみ、あるいは3次元以上の変換やたたみこみなどに利用できる。一般に、分離可能な変換は、入力値として一連の数値をとって出力として一連の新しい数値を定義する1次元変換を使用する。分離可能な変換は、2つのこのような1次元変換の合成を有する。第1の1次元変換は、一連の数値の各セットそれぞれについて計算され、新しい一連の数値のセットが生成される。第2の変換は、この新しい一連の数値のセットから対応する位置の数値を連続してとることにより得られる横方向の一連の数値について計算される。

上記それぞれの場合において、変換する必要のある数値は、オペランドのセグメント内に格納してもよい。その場合、数値が格納されるセグメントの位置は、数値が位置する列によって各行ごとに同じ方法で決定される（各オペランド内の数値は同じ行に属す）。変換は、専用命令を使用して行方向に、そしてSIMD命令によって行を横切る方向に並列に何度でも実行できる。

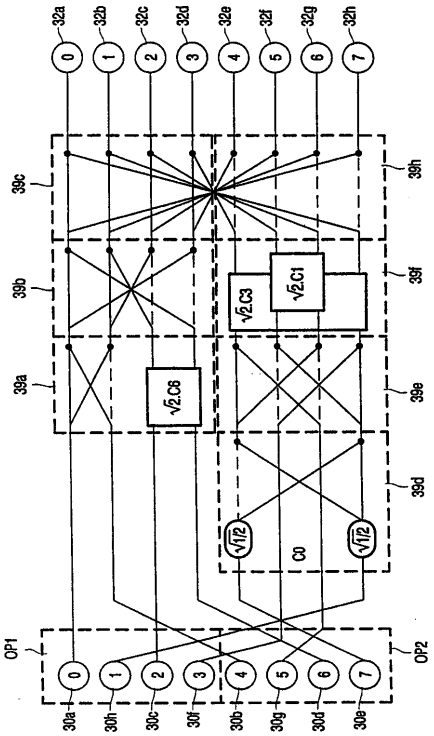
【図１】



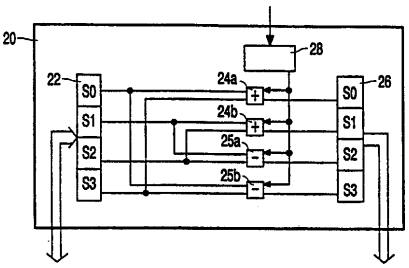
【図２】



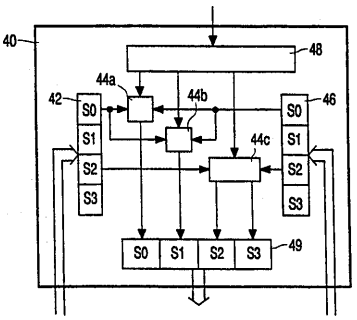
【図３】



【図４ａ】



【図４ｂ】



フロントページの続き

(72)発明者　スェスターマンス　フランシスカス　ダブリュ
オランダ国　５６５６　アーアー　アインドーフェン　プロフ　ホルストラーン　６

審査官　須田　勝巳

(56)参考文献　特開平０８－２４９２９３（ＪＰ，Ａ）
特開平０７－１４１３０４（ＪＰ，Ａ）
米国特許第０５６３８０６８（ＵＳ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 17/00 - 17/18
G06F 15/16