(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4623963号
(P4623963)
(45)発行日　平成23年2月2日(2011.2.2)　(24)登録日　平成22年11月12日(2010.11.12)

(51)Int.Cl.　F I
G06F 7/24 (2006.01)　G06F 7/24 M
G06F 17/10 (2006.01)　G06F 17/10 A

請求項の数 3　(全 46 頁)

(21)出願番号　特願2003-540797(P2003-540797)
(86)(22)出願日　平成14年10月28日(2002.10.28)
(65)公表番号　特表2005-508043(P2005-508043A)
(43)公表日　平成17年3月24日(2005.3.24)
(86)国際出願番号　PCT/US2002/034404
(87)国際公開番号　WO2003/038601
(87)国際公開日　平成15年5月8日(2003.5.8)
審査請求日　平成17年10月28日(2005.10.28)
(31)優先権主張番号　09/952,891
(32)優先日　平成13年10月29日(2001.10.29)
(33)優先権主張国　米国(US)
(31)優先権主張番号　10/280,612
(32)優先日　平成14年10月25日(2002.10.25)
(33)優先権主張国　米国(US)

前置審査

(73)特許権者　593096712
インテル　コーポレイション
アメリカ合衆国　９５０５２　カリフォルニア州　サンタ　クララ　ミッション　カレッジ　ブールバード　２２００
(74)代理人　100070150
弁理士　伊東　忠彦
(74)代理人　100091214
弁理士　大貫　進介
(74)代理人　100107766
弁理士　伊東　忠重

最終頁に続く

(54)【発明の名称】コンテンツデータを効率的にフィルタリング及び畳み込む方法及び装置

(57)【特許請求の範囲】
【請求項１】
プロセッサにより実行されるコンテンツデータを効率的にフィルタリング及び畳み込む方法であって、
前記プロセッサがデータシャッフル命令を実行することに応答して、マスクの値に従ってソースデータの選択された部分を、目的データ記憶装置内のデータが係数データ記憶装置内の係数と乗算されるように配置されるよう、整理するステップと、
前記プロセッサが乗加算命令を実行することに応答して、前記目的データ記憶装置内のバイトデータと前記係数データ記憶装置内のバイト係数との複数の積和ペアを生成するステップと、
前記プロセッサが隣接加算命令を実行することに応答して、１以上のデータ処理演算結果を構成するため、前記複数の積和ペアの隣接する積和ペアを加算するステップと、
前記１以上のデータ処理演算結果をデータ記憶装置に格納するステップと、
から構成され、
前記データシャッフル命令は、バイトシャッフル命令を含み、前記整理するステップは、前記ソースデータの選択された部分の個々のバイトを整理するバイトシャッフル処理を実行し、
前記バイトシャッフル命令はさらに、４バイトを有する各ダブルワードを、前記各ダブルワード内の４バイトの相対位置を維持することによって、整理できることを特徴とする方法。

【請求項２】
プロセッサにより実行されると、
データシャッフル命令の実行に応答して、マスクの値に従ってソースデータの選択された部分を、目的データ記憶装置内のデータが係数データ記憶装置内の係数と乗算されるように配置されるよう、整理するステップと、
乗加算命令の実行に応答して、前記目的データ記憶装置内のバイトデータと前記係数データ記憶装置内のバイト係数との複数の積和ペアを生成するステップと、
隣接加算命令の実行に応答して、１以上のデータ処理演算結果を構成するため、前記複数の積和ペアの隣接する積和ペアを加算するステップと、
前記１以上のデータ処理演算結果を格納するステップと、
から構成される方法を実行するプログラム命令を有するコンピュータ可読記憶媒体であって、
前記データシャッフル命令は、バイトシャッフル命令を含み、前記整理するステップは、前記ソースデータの選択された部分の個々のバイトを整理するバイトシャッフル処理を実行し、
前記バイトシャッフル命令はさらに、４バイトを有する各ダブルワードを、前記各ダブルワード内の４バイトの相対位置を維持することによって、整理できることを特徴とする記憶媒体。
【請求項３】
命令を実行する回路を有するプロセッサと、
前記プロセッサに接続され、命令シーケンスが格納される記憶装置と、
から構成される装置であって、
前記命令シーケンスは、前記プロセッサによる実行時に、
データ処理演算に従って、記憶装置内に構成されるように、係数を係数データ記憶装置にロードするステップと、
データシャッフル命令の実行に応答して、マスクの値に従ってソースデータの選択された部分を、目的データ記憶装置内のデータが前記係数データ記憶装置内の係数と乗算されるように配置されるよう、整理するステップと、
乗加算命令の実行に応答して、前記目的データ記憶装置内のバイトデータと前記係数データ記憶装置内のバイト係数との複数の積和ペアを生成するステップと、
隣接加算命令の実行に応答して、１以上のデータ処理演算結果を構成するため、前記複数の積和ペアの隣接する積和ペアを加算するステップと、
１以上のデータ処理演算結果を格納するステップと、
を前記プロセッサに実行させ、
前記データシャッフル命令は、バイトシャッフル命令を含み、前記整理するステップは、前記ソースデータの選択された部分の個々のバイトを整理するバイトシャッフル処理を実行し、
前記バイトシャッフル命令はさらに、４バイトを有する各ダブルワードを、前記各ダブルワード内の４バイトの相対位置を維持することによって、整理できることを特徴とする装置。
【発明の詳細な説明】
【発明の詳細な説明】
【０００１】
本特許出願は、２００１年１０月２９日に出願された米国特許出願第０９／９５２，８９１号「コンテンツデータの効率的フィルタリング及び畳み込み装置及び方法（An Apparatus And Method For Efficient Filtering And Convolution Of Content Data）」の一部継続出願である。
【０００２】
本特許出願は、２００２年１０月２５日に出願された同時係属中の米国特許出願第１０

／２８０，６１２号「ＳＩＭＤマージ命令による高速全探索動き（Ｆａｓｔ　Ｆｕｌｌ　Ｓｅａｒｃｈ　Ｍｏｔｉｏｎ　Ｗｉｔｈ　ＳＩＭＤ　Ｍｅｒｇｅ　Ｉｎｓｔｒｕｃｔｉｏｎ）」に関する。

［発明の技術分野］

本発明は、一般に、マイクロプロセッサ及びコンピュータシステムの分野に関する。より詳細には、本発明は、データの右方向平行シフトマージ（ｐａｒａｌｌｅｌ　ｓｈｉｆｔ　ｒｉｇｈｔ　ｍｅｒｇｅ）方法及び装置に関する。

［発明の背景］

プロセッサ技術の進歩により、このようなプロセッサを備えたマシーン上で実行される新しいソフトウェアコードが生成されている。一般に、ユーザは、使用しているソフトウェアのタイプに関わらず、コンピュータからより高いパフォーマンスを期待及び要求する。このような問題は、プロセッサ内部において実行されている命令及び処理のタイプから生じる可能性がある。あるタイプの処理には、必要とされる回路の処理及び／あるいはタイプの複雑さに基づき、その完了に多くの時間を要するものもある。このようなことから、複雑な処理のプロセッサ内部での実行方法を最適化するという動機付けが生じる。

【０００３】

メディアアプリケーションは、数十年もの間、マイクロプロセッサの発達を促進してきた。実際、近年における大部分の計算機の性能向上はメディアアプリケーションにより促進されてきたものである。娯楽性を高めた教育及び通信目的のため、重大な進歩は企業部門において見出されてきたが、上記のような性能の向上は主として消費者部門において起こってきたものである。にもかかわらず、これからのメディアアプリケーションには、さらに高い計算要件が要求されるであろう。この結果、将来のパーソナルコンピュータ（ＰＣ）では、使い安さだけでなくより充実したオーディオビジュアル機能が実現されるであろう。さらに、より重要なものとしては、計算機が通信と融合されるであろう。

【０００４】

従って、現在の計算機においては、コンテンツとして総称される音声及び映像データの再生だけでなく画像の表示も、ますます一般的なアプリケーションとなりつつある。フィルタリング及び畳み込み処理は、画像、音声及び映像データのようなコンテンツデータに対し最もよく実行される処理である。当業者には周知のように、フィルタリング及び相関計算では、データや係数の積を加える乗加算演算により計算される。２つのベクトルＡとＢの相関は、和Ｓの計算からなる。

【０００５】

【数１】

$$S[k]=\frac{1}{N}\sum_{i=0}^{N-1}a[i]\cdot b[i+k] \qquad \text{等式（1）}$$

これはしばしばｋ＝０により使われる。

【０００６】

【数２】

$$S[0]=\frac{1}{N}\sum_{i=0}^{N-1}a[i]\cdot b[i] \qquad \text{等式（2）}$$

ベクトルＶに適用されるＮタップフィルタｆの場合、計算される和Ｓは以下のようになる。

【０００７】

【数３】

$$S = \sum_{i=0}^{N-1} f[i] \cdot V[i] \qquad \text{等式（3）}$$

このような演算には大きな計算量を要する一方、例えば、単一命令多重データ（ＳＩＭＤ）レジスタのような様々なデータ格納装置を利用した効率的実現法を通じ利用可能なハイレベルのデータ並列処理を提供する。

【０００８】

フィルタリング処理の応用は、より広い範囲の画像及び映像処理タスク及び通信において見出すことができる。フィルタの利用例としては、ＭＰＥＧ（Ｍｏｔｉｏｎ　Ｐｉｃｔｕｒｅ　Ｅｘｐｅｒｔ　Ｇｒｏｕｐ）映像におけるブロックアーチファクトの低減、ノイズや音声の低減、透かし検出を向上させるための画素値からの透かしの抽出、スムージング、シャープニング、ノイズ低減、エッジ検出、画像または映像フレームサイズのスケーリングのための相関、サブピクセル動き予測のための映像フレームのアップサンプリング、音声信号の音質向上、及び通信における信号のパルス整形及び等化処理などが挙げられる。従って、畳み込み処理だけでなくフィルタリング処理もまた、画像、音声及び映像データを含むコンテンツの再生を提供する計算機とって重要なものである。

【０００９】

しかしながら、既存の方法及び指示はフィルタリングの通常必要とされるものを満足することを目的としたものであり、より広い範囲をカバーするものではない。実際、多くのアーキテクチャが様々なフィルタ長及びデータタイプに対する効率的なフィルタ計算手段をサポートしていない。さらに、レジスタ内及びレジスタ間における部分的なデータ転送に対する隣接する値の加算と共に、ＳＩＭＤレジスタのようなデータ記憶装置におけるデータオーダリングは、一般にサポートされていない。その結果、既存のアーキテクチャは不必要なデータタイプの変更を必要とし、それによって、命令あたりの処理数が最小化され、算術演算のためのデータ順序付けに要するクロックサイクル数を著しく増加させてしまう。

【００１０】

本発明が実施例を利用することにより示される。本発明は添付される図面に制限されるものではない。図面中、同一の参照記号は同一の要素を示している。

［詳細な説明］

データに対し右方向平行シフトマージを実行する方法及び装置が開示される。また、コンテンツデータの効率的なフィルタリング及び畳み込みを実行する方法及び装置が開示される。さらにまた、ＳＩＭＤマージ処理による高速全探索動き検出のための方法及び装置が開示される。ここで説明される実施例はマイクロプロセッサに関するものであるが、必ずしもそれに限定されるものではない。以下の実施例はプロセッサに関し説明されるが、他の実施例では、他のタイプの集積回路及び論理装置に適用することもできる。本発明の同様なテクニック及び教示は、より高いパイプラインスループット及び性能を享受しうる他のタイプの回路あるいは半導体デバイスに容易に適用することができる。本発明の教示は、データ操作を実行する任意のプロセッサあるいはマシーンに適用可能である。しかしながら、本発明は、２５６ビット、１２８ビット、６４ビット、３２ビットあるいは１６ビットデータ処理を実行するプロセッサあるいはマシーンに限定されるものでなく、データの右方向シフトマージを必要する任意のプロセッサ及びマシーンに適用することができる。

【００１１】

以下の記述では、説明のため、本発明の完全な理解を提供するため様々な具体的詳細が与えられる。本発明の実践に対し、これら具体的詳細が必ずしも必要でないということは当業者には認識されるであろう。また、周知の電気構造及び回路は、本発明を不必要に不

明瞭にしないよう詳細には与えられていない。さらに、以下の説明は実施例を与えるものであり、添付される図面は例示のため様々な実施例を示している。しかしながら、これらの実施例は限定のためのものと解釈されるべきでない。これらの実施例は、本発明のすべての可能な実現を包括的に列挙するものでなく、単に本発明の例を提供することを目的としている。

【００１２】

一実施例において、本発明による方法は、マシーン実行可能な命令により実現される。これらの命令により、プログラム可能な汎用あるいは特定用途向けプロセッサは本発明の各ステップを実行する。あるいは、本発明の各ステップは、これらのステップを実行する配線論理を含む特定のハードウェア要素により実行されてもよいし、あるいはプログラムされたコンピュータ構成要素及びカスタムハードウェア構成要素による任意の組み合わせにより実行されてもよい。

【００１３】

本発明は、これに従う処理を実行するようコンピュータ（あるいは他の電子装置）をプログラムするのに利用される命令を有するマシーンまたはコンピュータ読み出し可能な媒体を備えたコンピュータプログラムプロダクツまたはソフトウェアとして与えることができる。そのようなソフトウェアはシステム内部のメモリに格納することができる。同様に、そのコードはネットワークまたは他のコンピュータ読み出し可能な媒体を介し配信されうる。コンピュータ読み出し可能な媒体には、以下に限定されるものではないが、フロッピーディスク（登録商標）、光ディスク、ＣＤ（Ｃｏｍｐａｃｔ　Ｄｉｓｃ）、ＣＤ－ＲＯＭ（ＣＤ　Ｒｅａｄ－Ｏｎｌｙ　Ｍｅｍｏｒｙ）、光磁気ディスク、ＲＯＭ（Ｒｅａｄ－Ｏｎｌｙ　Ｍｅｍｏｒｙ）、ＲＡＭ（Ｒａｎｄｏｍ　Ａｃｃｅｓｓ　Ｍｅｍｏｒｙ）、ＥＰＲＯＭ（Ｅｒａｓａｂｌｅ　Ｐｒｏｇｒａｍｍａｂｌｅ　Ｒｅａｄ－Ｏｎｌｙ　Ｍｅｍｏｒｙ）、ＥＥＰＲＯＭ（Ｅｌｅｃｔｒｉｃａｌｌｙ　Ｅｒａｓａｂｌｅ　Ｐｒｏｇｒａｍｍａｂｌｅ　Ｒｅａｄ－Ｏｎｌｙ　Ｍｅｍｏｒｙ）、磁気あるいは光カード、フラッシュメモリ、インターネット上の送信などが含まれる。

【００１４】

従って、コンピュータ読み出し可能な媒体には、マシーン（例えば、コンピュータ）による読み出し可能な形態により電子的命令あるいは情報の格納または送信に適した任意のタイプのメディア／マシーン読み出し可能な媒体が含まれる。さらに、本発明はまた、コンピュータプログラムプロダクツとしてダウンロード可能であってもよい。その場合、プログラムはリモートコンピュータ（例えば、サーバ）からリクエストコンピュータ（例えば、クライアント）に転送される。プログラムの転送は、電子、光、音響あるいは搬送波で実現される他の形態のデータ信号、あるいは通信リンク（例えば、モデム、ネットワーク接続など）を介した他の伝搬媒体により実行されてもよい。

【００１５】

今日のプロセッサでは、様々なコード及び命令の処理及び実行に多くの実行ユニットが利用されている。命令の中には即座に完了するものがある一方、膨大なクロックサイクルを要する命令もあるので、必ずしもすべての命令が等しく生成されるとは限らない。命令のスループットが速くなるほど、プロセッサの全体的なパフォーマンスはより向上する。従って、できる限り多くの命令が実行されることが望ましい。しかしながら、大きな複雑さを有し、多くの実行時間及びプロセッサリソースを要する命令もある。例えば、浮動小数点命令、ロード／ストア処理、データ転送などが挙げられる。

【００１６】

ますます多くのコンピュータシステムがインターネットやマルチメディアアプリケーションにおいて利用されるに従い、追加的なプロセッササポートがこれまで導入されてきた。例えば、単一命令多重データ（ＳＩＭＤ）整数／浮動小数点命令やストリーミングＳＩＭＤエクステンション（ＳＳＥ）は、特定のプログラムタスクの実行に要する全体の命令数を減少させる命令である。これらの命令は、複数のデータ要素に対し並列処理を行うことにより、ソフトウェアパフォーマンスの高速化を可能にする。これにより、映像、音声

、及び画像／フォト処理を含む広範なアプリケーションにおいてパフォーマンスの向上を達成することが可能となる。通常、マイクロプロセッサや類似の論理回路におけるＳＩＭＤ命令の実現には多くの発行が伴う。さらに、ＳＩＭＤ処理の複雑さはしばしば、正確なデータ処理及び操作のための追加的回路の必要性を生じさせる。

【００１７】

本発明の実施例は、ＳＩＭＤに関するハードウェアを利用するアルゴリズムとして、右方向平行シフトの実現方法を提供する。一例として、当該アルゴリズムは、所望の数のデータセグメントをあるオペランドから第２のオペランドの最上位サイドに右方向シフトし、同数のデータセグメントが第２のオペランドの最下位サイドにシフトするというコンセプトに基づいている。概念的には、この右方向シフトマージ処理では、データの２つのブロックを１つのブロックとしてマージし、新たなデータパターンを形成するために、データセグメントを所望の位置で揃えるようマージされたブロックのシフトが実行される。従って、本発明による右方向シフトマージアルゴリズムの実施例は、全体のパフォーマンスを大きく損なうことなく効率的にＳＩＭＤ処理をサポートするプロセッサにおいて実現可能である。

コンピュータアーキテクチャ

図１は、本発明の一実施例が実現されうるコンピュータシステム１００を示す。コンピュータシステム１００は、情報を通信するためのバス１０１と、バス１０１に接続され、情報を処理するプロセッサ１０９から構成される。コンピュータシステム１００はまた、バス１０１に接続され、プロセッサ１０９のための情報や命令を格納するメモリサブシステム１０４～１０７を備える。

【００１８】

プロセッサ１０９は、実行ユニット１３０、レジスタファイル２００、キャッシュメモリ１６０、デコーダ１６５及び内部バス１７０から構成される。ｋ多ッ主メモリ１６０は、実行ユニット１３０に接続され、プロセッサ１０９のために頻繁に及び／または最近使用された情報を格納する。レジスタファイル２００は、プロセッサ１０９における情報を格納し、内部バス１７０を介し実行ユニット１３０に接続される。本発明の一実施例では、レジスタファイル２００は、マルチメディア情報を格納するＳＩＭＤレジスタのような複数のマルチメディアレジスタを備える。一実施例では、各マルチメディアレジスタは、１２８ビットまでのパケットデータを格納する。マルチメディアレジスタは、専用マルチメディアレジスタであってもよいし、あるいはマルチメディア情報や他の情報の格納に利用されるレジスタであってもよい。一実施例では、マルチメディアレジスタは、マルチメディア処理の実行時にはマルチメディア情報を格納し、浮動小数点演算の実行時には浮動小数点データを格納する。

【００１９】

実行ユニット１３０は、Ｐａｃｋｅｄ命令セット１４０に含まれるプロセッサ１０９により受信された命令セットに従いＰａｃｋｅｄデータを処理する。実行ユニット１３０はまた、汎用プロセッサにおいて実現される命令に従って、スカラーデータを処理する。プロセッサ１０９は、Ｐｅｎｔｉｕｍ（登録商標）マイクロプロセッサ命令セットとＰａｃｋｅｄ命令セット１４０をサポートすることができる。Ｐｅｎｔｉｕｍ（登録商標）マイクロプロセッサ命令セットのような標準的なマイクロプロセッサ命令セットにＰａｃｋｅｄ命令セット１４０を含めることにより、Ｐａｃｋｅｄデータ命令を（標準的なマクロプロセッサ命令セットのために以前に書かれた）既存のソフトウェアに容易に内蔵することができる。ＰｏｗｅｒＰＣ（商標）やＡｌｐｈａ（商標）プロセッサの命令セットのような他の標準的な命令セットが、説明される本発明に従い使用されてもよい。（Ｐｅｎｔｉｕｍ（登録商標）は、インテルコーポレーションの登録商標である。ＰｏｗｅｒＰＣ（商標）は、ＩＢＭ、アップルコンピュータ及びモトローラの商標である。Ａｌｐｈａ（商標）は、デジタルイクイップメントコーポレーションの商標である。）

一実施例では、Ｐａｃｋｅｄ命令セット１４０は、（以下でさらなる詳細が示されるように）転送データ（ＭＯＶＤ）処理１４３と、データ記憶装置内のデータを構成するデー

タシャッフル（ＰＳＨＵＦＤ）処理１４５のための命令を含む。符号なし第１ソースレジスタと符号付き第２ソースレジスタに対するＰａｃｋｅｄ乗算及び加算（ＰＭＡＤＤＵＳＢＷ処理１４７）。符号なし第１ソースレジスタと符号なし第２ソースレジスタに対する乗加算を実行するＰａｃｋｅｄ乗加算処理（ＰＭＡＤＤＵＵＢＷ処理１４９）。符号付き第１ソースレジスタと符号付き第２ソースレジスタに対するＰａｃｋｅｄ乗加算処理（ＰＭＡＤＤＳＳＢＷ処理１５１）と、１６ビットデータを含む符号付き第１及び第２ソースレジスタに対するＰａｃｋｅｄ乗加算処理（ＰＭＡＤＤＷＤ処理１５３）。最終的に、Ｐａｃｋｅｄ命令セットには、隣接バイトを加える隣接加算命令（ＰＡＡＤＤＮＢ処理１５５）、隣接ワードを加える隣接加算命令（ＰＡＡＤＤＮＷＤ処理１５７）、隣接ダブルワードを加える隣接加算命令（ＰＡＡＤＤＮＤＷＤ処理１５９）、２つのワード値を加える隣接加算命令（ＰＡＡＤＤＷＤ処理１６１）、１６ビットの結果を生成するための２つのワードを加える隣接加算命令（ＰＡＡＤＤＮＷＷ処理１６３）、クアドワード結果を生成するため２つのクアドワードを加える隣接加算命令（ＰＡＡＤＤＮＤＤ処理１６５）、及びレジスタマージ処理１６７が含まれる。

【００２０】

Ｐａｃｋｅｄ命令セット１４０を汎用プロセッサ１０９の命令セットに、命令を実行する関連する回路と共に含めることによって、多くの既存のマルチメディアアプリケーションにより使われる処理が汎用プロセッサのＰａｃｋｅｄデータを利用することにより実行されてもよい。従って、Ｐａｃｋｅｄデータを処理するプロセッサのデータバスの最大幅を利用することにより、多くのマルチメディアアプリケーションは高速化され、より効率的に実行される。これにより、データのより小さなユニットをプロセッサのデータバスに転送し、一度にデータあたり複数の処理を実行させる必要がなくなる。

【００２１】

図１を参照するに、本発明のコンピュータシステム１００は、モニタのような表示装置１２１を含んでいてもよい。表示装置１２１は、フレームバッファのような中間装置を含んでいてもよい。コンピュータシステム１００はまた、キーボードのような入力装置１２２と、マウス、トラックボールまたはトラックパッドのようなカーソル制御１２３を含んでいてもよい。表示装置１２１、入力装置１２２及びカーソル制御１２３は、バス１０１に接続される。コンピュータシステム１００はまた、コンピュータシステム１００がローカルエリアネットワーク（ＬＡＮ）やワイドエリアネットワーク（ＷＡＮ）の一部となるようネットワークコネクタ１２４を含んでもよい。

【００２２】

さらに、コンピュータシステム１００は、音声認識のための音声入力を記録するためのマイクロフォンに接続されるオーディオデジタイザのような音声記録及び／あるいは再生装置１２５に接続することができる。コンピュータシステム１００はまた、映像のキャプチャに利用可能な映像デジタル化装置１２６、プリンタのようなハードコピー装置１２７、及びＣＤ－ＲＯＭ装置１２８を備えてもよい。これらの装置１２４～１２８はまた、バス１０１に接続される。

プロセッサ

図２は、プロセッサ１０９の詳細な図を示す。プロセッサ１０９は、ＢｉＣＭＯＳ、ＣＭＯＳ及びＮＭＯＳのような多くのプロセス技術の何れかを利用した複数の基板において実現することができる。プロセッサ１０９は、プロセッサ１０９により使われる制御信号及びデータの復号化を行うデコーダ２０２を備える。その後、データは内部バス２０５を介しレジスタファイル２００に格納される。問題を明瞭にするため、一例となるレジスタは特定のタイプの回路に限定されるべきでない。一例となるレジスタは、データの格納及び供給、並びにここで説明される機能の実行が可能であることが必要である。

【００２３】

データタイプに応じて、データは、整数レジスタ２０１、レジスタ２０９、ステータスレジスタ（ｓｔａｔｕｓ　ｒｅｇｉｓｔｅｒ）２０８、または命令ポインタレジスタ２１１に格納することができる。例えば、浮動小数点レジスタのような他のレジスタは、レジ

スタファイル２０４に含めることができる。一実施例では、整数レ擬すタ２０１は３２ビット整数データを格納する。一実施例では、レジスタ２０９は、例えば、Ｐａｃｋｅｄデータを含むＳＩＭＤレジスタのような８つのマルチメディアレジスタ$R_0$２１２ａ～$R_7$２１２ｈを有する。レジスタ２０９の各レジスタは、１２８ビット長である。$R_1$２１２ａ、$R_2$２１２ｂ及び$R_3$２１２ｃは、レジスタ２０９の各レジスタの一例である。レジスタ２０９のあるレジスタの３２ビットを整数レジスタ２０１の整数レジスタに転送することができる。同様に、整数レジスタの値をレジスタ２０９のあるレジスタの３２ビットに転送することができる。

【００２４】

ステータスレジスタ２０８は、プロセッサ１０９の状態を示す。命令ポインタレジスタ２１１は、実行されるべき次の命令のアドレスを格納している。整数レジスタ２０１、レジスタ２０９、ステータスレジスタ２０８及び命令ポインタレジスタ２１１はすべて内部バス２０５に接続される。任意の追加的レジスタがまた内部バス２０５に接続されるであろう。

【００２５】

他の実施例では、これらのレジスタのいくつかは、２つのデータタイプに利用することができる。例えば、レジスタ２０９と整数レジスタ２０１は、各レジスタが整数データあるいはＰａｃｋｅｄデータのどちらかを格納できるよう一体化されてもよい。他の実施例では、レジスタ２０９は浮動小数点レジスタとして利用することができる。この実施例では、Ｐａｃｋｅｄデータはレジスタ２０９あるいは浮動小数点データに格納することができる。一実施例では、一体化されたレジスタは１２８ビット長を有し、整数は１２８ビットとして表現される。この実施例では、Ｐａｃｋｅｄデータ及び整数データの格納において、レジスタはこれら２つのデータタイプを区別する必要はない。

【００２６】

機能ユニット２０３は、プロセッサ１０９により実行される処理を行う。そのような処理には、シフト、加算、減算及び乗算などが含まれうる。機能ユニット２０３は内部バス２０５に接続される。キャッシュ１６０は、プロセッサ１０９の選択的ユニットであり、例えば、メインメモリ１０４からのデータ及び／あるいは制御信号をキャッシュするのに利用される。キャッシュ１６０は、デコーダ２０２に接続され、制御信号２０７を受信するよう接続される。

データ及び記録フォーマット

図３は、Ｐａｃｋｅｄバイト２２１、Ｐａｃｋｅｄワード２２２及びＰａｃｋｅｄダブルワード（ｄｗｏｒｄ）２２３の３つのＰａｃｋｅｄデータタイプを示す。Ｐａｃｋｅｄバイト２２１は、１６のＰａｃｋｅｄバイトデータ要素を含む１２８ビット長である。一般に、データ要素とは、同じデータ長を有する他のデータ要素と共に１つのレジスタ（あるいはメモリ位置）に格納される個々のデータである。Ｐａｃｋｅｄデータ系列では、レジスタに格納されるデータ要素数は、データ要素のビット長により分割される１２８ビットである。

【００２７】

Ｐａｃｋｅｄワード２２２は、１２８ビット長であり、８つのＰａｃｋｅｄワードデータ要素を含む。各Ｐａｃｋｅｄワードは１６ビットの情報を含んでいる。ｐａｃｋｅｄダブルワード２２３は、１２８ビット長であり、４ついのＰａｃｋｅｄダブルワードデータ要素を含んでいる。各Ｐａｃｋｅｄ　ダブルワードデータ要素は、３２ビットの情報を含んでいる。Ｐａｃｋｅｄクアドワードは、１２８ビット長で、２つのＰａｃｋｅｄクアドワードデータ要素を含んでいる。

【００２８】

図４Ａ～４Ｃは、本発明の一実施例によるイン・レジスタ（ｉｎ－ｒｅｇｉｓｔｅｒ）Ｐａｃｋｅｄデータ記録表現を示す。図４Ａに示されるように、符号なしＰａｃｋｅｄバイトイン・レジスタ表現３１０により、符号なしＰａｃｋｅｄバイト２０１のマルチメディアレジスタ２０９の１つへの記録が示される。各バイトデータ要素の情報は、第０バイ

トには第７ビットから第０ビットに、第１バイトには第１５ビットから第８ビットに、第２バイトには第２３ビットから第１６ビットに、最後に、第１５バイトには第１２８ビットから第１２７ビットから第１２０ビットにそれぞれ格納される。

【００２９】

これにより、利用可能なすべてのビットがレジスタにおいて使用される。この記録配置は、プロセッサの記憶効率を高めるものである。さらに、１６のデータ要素がアクセスされることにより、１つの処理がこれら１６のデータ要素に同時に実行することができる。符号付きＰａｃｋｅｄバイトイン・レジスタ表現３１１により、符号付きＰａｃｋｅｄバイト２１１の記録が示される。ここで、すべてのバイトデータ要素の第８ビットは符号表示に使われる。

【００３０】

図４Ｂに示されるように、符号なしＰａｃｋｅｄワードイン・レジスタ表現３１２により、第７ワードから第０ワードまでがどのようにマルチメディアレジスタ２０９のレジスタに格納されるか示されている。符号付きＰａｃｋｅｄワードイン・レジスタ表現３１３は、符号なしＰａｃｋｅｄワードイン・レジスタ表現３１２と同様である。ここで、各ワードデータ要素の第１６ビットは符号表示に使われる。図４Ｃに示されるように、符号なしＰａｃｋｅｄダブルワードイン・レジスタ表現３１４により、マルチメディアレジスタ２０９がどのように２つのダブルワードデータ要素を格納するか示されている。符号付きＰａｃｋｅｄダブルワードイン・レジスタ表現３１５は、符号なしＰａｃｋｅｄダブルワードイン・レジスタ表現３１４と同様である。ここで、必要な符号ビットは、ダブルワードデータ要素の第３２ビットである。

【００３１】

本発明のより教示されるように、コンテンツデータの効率的フィルタリング及び畳み込みは、データ及びフィルタ／畳み込み係数によるデータソース装置のロードにより開始される。多くの場合、例えば、単一命令多重データ（ＳＩＭＤ）レジスタのようなデータ記憶装置内のデータあるいは係数の順序は、算術計算が実行される前に変更を要する。従って、効率的なフィルタ計算及び畳み込みには、適切な算術命令だけでなく、計算の実行に要する効率的なデータ構造化方法が必要とされる。

【００３２】

例えば、バックグラウンド部において記号を利用して、例えば、Ｓ［Ｉ］により与えられる画素Ｉの値を置換することにより画像がフィルタリングされる。画素Ｉの何れかのサイドにおける画素値がＳ［Ｉ］のフィルタリング計算において使用される。同様に、画素Ｉ＋１の何れかのサイドにおける画素が、ＳＤ［Ｉ＋１］の計算に必要とされる。これにより、ＳＩＭＤレジスタにおける１より多い画素のフィルタリング結果を計算するために、データが複製され、ＳＩＭＤレジスタに置かれる。

【００３３】

しかしながら、既存のコンピュータアーキテクチャでは、アーキテクチャ内の適切なすべてのデータサイズに対する効率的なデータ配置方法が欠落している。従って、図５に示されるように、本発明は、任意のサイズのデータを効率的に順序付けするバイトシャッフル命令（ＰＳＨＵＦＢ）１４５を備える。このバイトシャッフル処理１４５では、シャッフル処理中にバイトの相対位置をより大きなデータ内に維持することにより、バイトより大きなデータサイズの順序付けが行われる。さらに、バイトシャッフル処理１４５では、ＳＩＭＤレジスタ内のデータの相対位置の変更、及びデータの複製もまた可能である。

【００３４】

図５を再度参照するに、図５は、３つの係数を有するフィルタに対するバイトシャッフル処理１４５の一例が示されている。従来技術を利用すると、フィルタ係数（図示せず）が３つの画素に適用され、その後このフィルタ係数が他の画素に移動され、再び適用される。しかしながら、これらの処理を並列に実行するために、本発明は、データの配置に新たな命令を導入する。従って、図５に示されるように、データ４０４は目的データ記憶装置（ｄｅｓｔｉｎａｔｉｏｎ　ｄａｔａ　ｓｔｏｒａｇｅ　ｄｅｖｉｃｅ）４０６内で構

成される。一実施例では、目的データ記憶装置４０６は、各データ要素を格納するアドレスを特定するのにマスク４０２を利用したソースデータ記憶装置（ｓｏｕｒｃｅ　ｄａｔａ　ｓｔｏｒａｇｅ　ｄｅｖｉｃｅ）４０４である。一実施例では、マスクの配置は、例えば、フィルタリング処理、畳み込み処理などを含む所望のデータ処理に基づく。

【００３５】

従って、マスク４０２を利用することにより、係数と共にデータ４０６は並列に処理することが可能になる。上述の実施例では、ソースデータ記憶装置４０４は、初期的に１６の８ビット画素を格納する１２８ビットＳＩＭＤレジスタである。さらに、３つの係数による画素フィルタが利用されるとき、第４の係数は０に設定される。一実施例では、ソースデータ記憶装置４０４内のデータ要素数に応じて、ソースレジスタ４０４は目的データ記憶装置あるいはレジスタとして利用され、それにより、一般に必要とされるよりレジスタ数を減少させることができる。さらに、ソースデータ記憶装置４０４内の上書きされたデータは、メモリから他のレジスタに再ロードされる。さらに、所望のように各データを目的データ記憶装置４０６内で構成することにより、複数のレジスタがソースデータ記憶装置４０４として利用されてもよい。

【００３６】

データ要素と係数の順序付けが完了したら、データと対応する係数がデータ処理に従い処理される必要がある。様々なフィルタ係数及びデータサイズを利用することにより、フィルタ計算と畳み込み計算には様々な精度による処理が必要とされる。最も基本的なフィルタ処理は、２つの数のペアの乗算と、それらの加算である。この処理は、乗加算命令と呼ばれる。

【００３７】

しかしながら、既知のコンピュータアーキテクチャでは、符号付きまたは符号なし係数を利用した複数の配列あるいはフィルタ長及びデータサイズに対する効率的な乗加算をサポートしていない。さらに、バイト演算もサポートされていない。その結果、従来技術によるコンピュータアーキテクチャは、Ｕｎｐａｃｋ命令を利用することにより１６ビットデータを変換しなければならない。一般に、これらのコンピュータアーキテクチャは、異なるレジスタにある１６ビットデータの積を計算し、隣接する積を加え合わせ、３２ビットの結果を与える乗加算処理をサポートしている。この解決策は１６ビットの精度を要するデータのフィルタ係数には受け入れられるが、（画像や映像において一般的な）８ビットデータの８ビットフィルタ係数に対しては、命令及びデータレベルの並列化は無駄である。

【００３８】

図６を参照するに、図６は、第１ソースレジスタ４５２と第２ソースレジスタ４５４を示す。一実施例では、第１及び第２ソースレジスタは、例えば、１２８ビットＩｎｔｅｌ（登録商標）ＳＳＥ２　ＸＭＭレジスタのようなＮビット長ＳＩＭＤレジスタである。このようなレジスタで実現される乗加算命令は、２つの画素ベクトル４５２と４５４に対して以下の結果を与え、目的レジスタ４５６内に格納される。従って、実施例はＰＭＡＤＤＵＳＢＷ処理１４７（図１）と呼ばれる８ビットバイト－１６ワード乗加算命令（８－ｂｉｔ　ｂｙｔｅ　ｔｏ　１６　ｗｏｒｄ　ｍｕｌｔｉｐｌｙ－ａｃｃｕｍｕｌａｔｅ　ｉｎｓｔｒｕｃｔｉｏｎ）を示す。ここで、命令中の「Ｕ」と「Ｓ」はそれぞれ符号なしと符号付きバイトを表している。ソースレジスタの一方におけるバイトは符号付きであり、もう一方におけるバイトは符号なしである。

【００３９】

本発明の一実施例では、符号なしデータを有するレジスタは、目的及び１６乗加算結果である。この選択は、大部分の実行において、データは符号なし、係数は符号付きであるという理由による。データは以降の計算において必要とされる確率は低いので、データを上書きすることが好ましい。図１に示されるような追加的バイト乗加算命令は、両方のレジスタにおける符号なしバイトに対するＰＭＡＤＤＵＵＢＷ処理１４９と、両方のソースレジスタにおける符号付きバイトに対するＰＭＡＤＤＳＳＢＷ処理１５１である。この乗

加算命令は、３２ビット符号付き積の生成のために、１６ビット符号付きワードのペアに適用されるＰＭＡＤＤＷＤ命令１５３により完了される。

【００４０】

一般的に、フィルタリング処理では、第２ベクトルはフィルタ係数を含んでいる。従って、ＸＭＭレジスタを用意するため、係数がレジスタの一部にロードされ、シャッフル命令１４５を使ってレジスタの残りの部分にコピーするようにしてもよい。例えば、図７Ａに示されるように、例えば、ＸＭＭ１２８ビットレジスタのような係数データ記憶装置５０２は、データロード命令の実行に応答して、３つの係数により初期ロードされる。しかしながら、フィルタ係数はデータ処理以前にメモリにおいて構成されるようにしてもよいということは当業者には理解されるであろう。さらに、フィルタリング処理以前に、係数はメモリ内の係数の構成に基づき図７Ｂに示されるように初期ロードされてもよい。

【００４１】

さらに、係数レジスタ５０２には、符号付きあるいは符号なしとして符号化が可能なフィルタ係数Ｆ３、Ｆ２及びＦ１が含まれる。係数レジスタ５０２がロードされると、既存の命令ＰＳＨＵＦＤを使って、図７Ｂに示されるような結果を得るために、係数レジスタの残り部分内のフィルタ係数をコピーする。図７Ｂに示されるように、係数レジスタ５０４には、データ処理の並列実行に要するシャッフルされた係数が含まれる。当業者に知られるように、３つの係数を含むフィルタは画像処理アルゴリズムではよく知られている。しかしながら、ＪＰＥＧ２０００のようなフィルタリング処理では、９つの１６ビット係数と７つの１６ビット係数が利用される。従って、このような係数の処理は係数レジスタの能力を超えるものであり、部分的にフィルタリングされた結果が生じることとなる。各係数を使って最終的な結果が得られるまで、処理が続けられる。

【００４２】

図７Ｃを参照するに、図７Ｃは、図５に示されるようなソースレジスタ４０４に初期的に含まれ、目的レジスタ４０６においてシャッフルされたソースレジスタ５０６の画素値の配置を示す。データ処理の実行に応答して、ＰＭＡＤＤＵＳＢＷ命令を使って、目的レジスタ５１０に格納されるにより２つの乗算の和が計算される。しかしながら、計算を完了させ、選ばれたデータ処理に対するデータ処理結果を生成するため、目的レジスタ５１０の隣接する積和ペアを加算する必要がある。

【００４３】

従って、乗加算命令の和が一般的である２画素より長い場合、個々の和が加算される必要がある。しかしながら、既存のコンピュータアーキテクチャでは、隣接する和が同じ目的レジスタにあるということから、隣接する和を加える効率的な方法は提供されていない。従って、本発明は、隣接加算命令を利用している。その結果が図８Ａ～図８Ｄに示されている。

【００４４】

図８Ａを参照するに、図８Ａは、３２ビット和を与えるために、２つの隣接する１６ビット値の加算（ＰＡＤＤＤ２ＷＤ処理１５７）に従う目的レジスタ５５２を示す。さらに、図８Ａは、４バイトの積の３２ビット和を与えるために加算される乗加算命令の２つの隣接する１６ビットの結果を示す。図８Ｂは、３２ビット和を与えるために４つの隣接する１６ビット値を加算する隣接加算命令（ＰＡＡＤＤＤ４ＷＤ処理１５７）を示す。さらに、バイト乗加算命令の４つの隣接する１６ビットの結果が、８バイトの積の３２ビット和を与えるために加算される。図８Ｃは、３２ビット和を与えるために８つの隣接１６ビット値を加算する隣接加算命令（ＰＡＡＤＤ８ＷＤ処理１５７）を示す。さらに、この例は、１６倍との積の３２ビット和を与えるために加算されるバイト乗加算処理の８つの隣接する１６ビットの結果を示す。

【００４５】

隣接加算処理を実行するための命令の選択は、和（Ｎ）におけるターン（ｔｕｒｎ）数に基づく。例えば、図７Ａから図７Ｃに示されるような３タップフィルタを利用することにより、第１命令（ＰＡＡＤＤ２ＷＤ処理１５７）は図８Ｄに示されるような以下の結果

を取得する。しかしながら、２つの１６ビット画素ベクトル（例えば、マクロブロックの第１ライン）間の相関に対して、図８Ｃに示されるような最終命令（ＰＡＡＤＤ８ＷＤ処理１５７）が利用される。このような処理は、ＳＩＭＤレジスタのサイズが大きくなるに従い、効率的な実行のためにますます重要となってきている。このような処理がなければ、多くの追加的な命令が必要となる。

【００４６】

さらに、本発明により示されるように、隣接加算命令セットは、加算可能な隣接値の数と共通のデータタイプを広範にサポートする。一実施例において、隣接する１６ビット値の加算は、２つの隣接する値を加算すること（Ｎ＝２）から開始され、加算対象の数を４（Ｎ＝４）、それから８（Ｎ＝８）、そしてレジスタの合計まで倍加する範囲を有する命令セットを含む。１６ビットの隣接加算の和のデータサイズは３２ビットである。他の実施例では、隣接する１６ビットの値が３２ビットの和となるよう加算される（ＰＡＡＤＤＷＤ処理１６１）。

【００４７】

この実施例では、１６ビットのデータサイズによる他の命令は含まれない。なぜなら、３２ビットの入力による隣接加算命令を使って、１６ビットの入力による命令によって生成される和が加算される。どちらの実施例においても、２つの隣接する値を加算すること（Ｎ＝２）から開始され、加算対象の数を４（Ｎ＝４）、それから８（Ｎ＝８）、そしてレジスタの合計まで倍加する範囲を有する３２ビット隣接加算命令セット（ＰＡＡＤＤＮＤＷＤ処理１５９）が含まれる。３２ビット隣接加算の和のデータサイズは３２ビットである。いくつかのケースでは、その結果はレジスタを満たさない。例えば、図８Ａから図８Ｃに示されるような命令である３つの相異なる隣接加算により、４、２及び１３２ビットの結果がもたらされる。一実施例では、これらの結果は目的データ記憶装置の下位及び最下位部分に格納される。

【００４８】

従って、図８Ｂに示されるように、２つの３２ビットの結果がある場合、この結果は下位の６４ビットに格納される。図８Ｃに示されるように、３２ビットの結果が１つである場合、この結果は下位の３２ビットに格納される。当業者により認識されるように、アプリケーションの中には隣接バイトの和を利用するものもある。本発明は、１６ビットワードを与える２つの隣接する符号付きバイトを加算する命令（ＰＡＡＤＤＮＢ処理１５５）と、１６ビットワード結果を与える２つの隣接する符号なしバイトを加算する命令によりバイトの隣接加算をサポートしている。２より多くの隣接バイトの加算を要するアプリケーションでは、適当な１６ビット隣接加算処理により２バイトの１６ビットの和が加算される。

【００４９】

データ処理結果が計算されると、次の処理はこの結果をメモリ装置に送ることからなる。上述の実施例によって示されるように、この結果は３２ビットの精度による符号化が可能である。従って、例えば、レジスタ全体に実行される右方向シフト論理処理（ＰＳＲＬＤＱ）や右方向シフトダブルクアドワード論理（ｓｈｉｆｔ　ｄｏｕｂｌｅ　ｑｕａｄ－ｗｏｒｄ　ｒｉｇｈｔ　ｌｏｇｉｃａｌ）と共に、上述のＭＯＶＤ処理１４３のようなダブルワードに関し実行されるシンプルな転送処理を利用することにより、結果がメモリに書き込まれる。さらに、すべての結果をメモリに書き込むには、第１のケース（図８Ａ）では４つのＭＯＶＤと３つのＰＳＲＬＤＱを必要とし、第２のケース（図８Ｂ）では２つのＭＯＶＤと１つのＰＳＲＬＤＱを必要とし、最後のケースでは図８Ｃに示されるように１つのＭＯＶＤが必要とされる。

【００５０】

しかしながら、図７Ｃに示されるように、隣接加算処理は並列に実行することが可能ではあるが、一般にフィルタリング計算では画像における次の画素が必要とされる。さらに、複数の画素がソースデータ記憶装置またはレジスタにロードされる必要がある。各時点で８つの画素をレジスタにロードすることを回避するために、この処理に２つの解決法が

提案される。一実施例では、本発明は、図９Ａに示されるように、レジスタマージ処理１６３を説明する。さらに、目的レジスタ６０６における画素Ａ１からＡ８を処理するために、画素Ａ７からＡ１が画素Ａ８により連結され、目的レジスタ６０６に画素Ａ８からＡ１が形成される。これにより、レジスタマージ処理は入力引数により与えられるバイト数を利用して、レジスタの選択を行う。

【００５１】

図９Ｂを参照するに、図９Ｂは、レジスタマージ処理の実行のための他の実施例を示す。初期的に、８つの画素が第１ソースレジスタ６０８（ＭＭ０）にロードされる。次に、後続の８つの画素が第２ソースレジスタ６１０（ＭＭ１）にロードされる。次に、置換処理が第２ソースレジスタ６１０に対し実行される。実行されると、レジスタ６１０が第３ソースレジスタ（ＭＭ２）６１２にコピーされる。次に、第１ソースレジスタ６０８が８ビット分右方向にシフトされる。さらに、第２ソースレジスタ６１０とマスクレジスタ６１４が、Ｐａｃｋｅｄ論理ＡＮＤ命令に従って合成され、第１ソースレジスタ６０８に格納される。

【００５２】

次に、論理ＯＲ演算が第２ソースレジスタ６１０と第１ソースレジスタ６０８に対し実行され、目的レジスタ６２０に結果が生成され、レジスタマージ処理が実行された。この処理は、第１ソースレジスタ６０８をシフトすることにより、図示されるように続けられる。次に、第２ソースレジスタ６１０がシフトされ、レジスタ６１２が生成される。次に、論理ＡＮＤ演算がマスクレジスタ６１４と第２ソースレジスタ６１２に対し実行され、結果が目的レジスタ６２２に格納される。最後に、Ｐａｃｋｅｄ　ＯＲ演算が第２ソースレジスタ６１２と第１ソースレジスタ６０８に対し実行され、目的レジスタ６２４に後続のレジスタマージ処理が生成される。本発明の教示を実現する手続き方法が説明される。

動作

図１０を参照するに、図１０は、例えば、図１及び図２に示されるようなコンピュータシステム１００におけるコンテンツデータの効率的フィルタリング及び畳み込みのための方法７００を示すブロック図を示す。ここで説明されるように、コンテンツデータとは、画像、音声及び映像データを意味する。さらに、本発明は、当業者に理解されるように、例えば、１２８ビットＩｎｔｅｌ（登録商標）アーキテクチャＳＳＥ２　ＭＭＸレジスタのようなデータレジスタを含むデジタルデータの格納が可能な様々な装置を有するデータ記憶装置に関し言及している。

【００５３】

図１０を再び参照するに、本発明による方法は処理ブロック７０２から開始され、データ処理が実行されているか判断される。ここで説明されるように、このデータ処理には、以下に限定されないが、画素データに関し実行される畳み込み及びフィルタリング処理が含まれる。これが実行されると、処理ブロック７０４が実行される。処理ブロック７０４では、データロード命令が実行される。このデータロード命令の実行に応答して、処理ブロック７０６において、入力データストリームデータが、例えば図２に示されるように、ソースデータ記憶装置２１２Ａと補助データ記憶装置２１２Ｂにロードされる。

【００５４】

処理ブロック７０８において、このデータ処理によりデータシャッフル命令が実行されたか判断される。データシャッフル命令の実行に応答して、処理ブロック７１０において、例えば、ソースデータ記憶装置２１２Ｂからのデータの選択部分が、目的データ記憶装置内で、あるいは係数データ記憶装置（図５を参照）内の係数配置に従って構成される。係数データ記憶装置内の係数が、所望のデータ処理計算に従って（例えば、図７Ａ及び図７Ｂに示されるように）構成される。一実施例では、フィルタリング処理以前に、係数はメモリ内で構成される。従って、シャッフルする必要なく、係数は係数データ記憶装置にロードすることができる（図７Ｂ参照）。

【００５５】

上述のように、図７Ａから図７Ｃに示されるように、データ処理に求められる並列計算

の実現にはデータ及び係数の順序付けが必要とされる。しかしながら、これらの係数はデータ処理前に既知となっているので、データ処理中係数をシャッフルする必要なくメモリにおいて構成されるように係数レジスタへのロードを可能にするために、係数はメモリ内で構成されるようにしてもよい。最後に、処理ブロック７２０において、ロードされたデータはデータ処理に従って処理され、１つ以上のデータ処理結果が生成される。生成されると、データ処理結果はメモリに書き込まれる。

【００５６】

図１１を参照するに、図１１は、データ処理に従ってデータを処理するための方法７２２を示すブロック図を示す。処理ブロック７２４において、データ処理が乗加算命令を実行したか判断される。乗加算命令の実行に応答して、処理ブロック７２６において、図７Ｃに示されるように、目的記憶装置におけるデータと係数データ記憶装置における係数の複数の積和ペアが生成される。次に、処理ブロック７２８において、データ処理が隣接加算命令を実行したか判断される。

【００５７】

隣接加算命令の実行に応答して、処理ブロック７３０において、目的データ記憶装置５１０（図７Ｃ）における隣接積和ペアが加算され、１つ以上のデータ処理結果が生成される（図８Ｄ参照）。しかしながら、ある実施例では、係数の個数が係数レジスタの容量を超える場合（処理ブロック７３２参照）、部分的なデータ処理結果が取得される。従って、係数（処理ブロック７３４）とデータ（処理ブロック７３６）の処理及び構成は、選択的な処理ブロック７３２から７３６に示されるように、最終的なデータ処理結果が得られるまで続けられる。そうでない場合、処理ブロック７３８において、１つ以上のデータ処理結果は格納される。最後に、処理ブロック７９０において、入力データストリームデータの処理が完了したか判断される。処理が完了すると、制御フローは処理ブロック７２０に戻り、方法７００は終了される。

【００５８】

図１２を参照するに、図１２は、追加的な入力データを処理するための追加的方法７４０を示すブロック図を示す。処理ブロック７４２において、ソースデータ記憶装置２１２Ａ内にアクセスされていないデータが存在するか判断される。ここで説明されるように、アクセスされていないデータとは、乗加算命令を実行するためデータ記憶装置においてシャッフルされていないソースデータ記憶装置２１２Ａ内のデータを意味する。データ記憶装置がアクセスされていないデータを含む場合、処理ブロック７４４において、データの一部が選択データとしてソースデータ記憶装置から選択される。選択されると、処理ブロック７８６が実行される。

【００５９】

そうでない場合、処理ブロック７４６において、ソースデータ記憶装置から１つ以上の未処理データ要素が選ばれると共に、補助データ記憶装置から１つ以上のデータ要素が選ばれる。ここで説明されるように、未処理データ要素とは、データ処理結果がまだ計算されていないデータ要素を意味している。次に、処理ブロック７８０において、レジスタマージ命令（図９Ａ及び図９Ｂ参照）が実行され、ソースデータ記憶装置の未処理データ要素と補助データ記憶装置から選ばれたデータ要素が連結され、選択されたデータが生成される。次に、処理ブロック７８２において、補助データ記憶装置からのデータがソースデータ記憶装置に転送される。

【００６０】

さらに、ソースデータ記憶装置は、すべて処理済であるため、もはや必要とされない。従って、未処理データを含むデータの補助記憶が利用され、ソースデータ記憶装置におけるデータが上書きされる。処理ブロック７８４において、補助データ記憶装置は、フィルタリング処理や畳み込み処理のような追加的なデータ処理に必要なメモリ装置からの入力データストリームデータによりロードされる。最後に、処理ブロック７８６において、選択されたデータが係数データ記憶装置（図５参照）における係数配置に従って、目的データ記憶装置内で構成される。これが実行されると、制御フローは選択されたデータの継続

処理のため、図１１に示されるように処理ブロック７９０に戻る。

【００６１】

図１３を参照するに、図１３は、未処理データ要素を選択するための追加的方法７４８を示す。処理ブロック７５０において、ソースデータ記憶装置に未処理データが含まれているか判断される。ソースデータ記憶装置におけるデータの各部分が処理済である場合、処理ブロック７７０が実行される。処理ブロック７７０において、選択されたデータとして機能するデータの一部が補助データ記憶装置から選ばれ、データ処理に従い処理される。

【００６２】

そうでない場合、処理ブロック７５２において、１つ以上の未処理データ要素がソースデータ記憶装置から選ばれる。最後に、処理ブロック７６６において、未処理データ要素の合計に従い補助データ記憶装置から追加的なデータ要素が選択され、選択されたデータが生成される。さらに、データ処理の実行前に、目的データ記憶装置におけるシャッフル処理のため選択されたデータは、フィルタ係数の個数に基づきデータ要素数に制限される。従って、このデータ要素の合計を利用して、レジスタマージ処理実行のため、補助データ記憶装置から選ばれるデータ要素数を決定するために、未処理データ要素数がデータ要素の合計から差し引かれる。

【００６３】

最後に、図１４を参照するに、図１４は、図１３に示されるように、処理ブロック７５２の未処理データ要素を選択するための追加的方法７５４を示す。処理ブロック７５６において、ソースデータ記憶装置からデータ要素が選ばれる。次に、処理ブロック７５８において、当該データ要素のデータ処理結果が計算されているかどうか判断ざれる。この結果が計算されている場合、選択されたデータ要素は破棄される。そうでない場合、処理ブロック７６０において、この選択されたデータ要素は未処理データ要素であり、格納される。次に、処理ブロック７６２において、未処理データ要素の合計がインクリメントされる。最後に、処理ブロック７６４において、ソースデータ記憶装置内の各データ要素が処理されるまで、処理ブロック７５６から７６２が繰り返される。

【００６４】

さらに、本発明の教示を使って、不要なデータタイプ変更が回避される。これにより、命令あたりのＳＩＭＤ処理数を最大化することができる。さらに、算術演算のためのデータ順序付けに要するクロックサイクル数を大幅に減らすことができる。従って、表１は、本発明のより説明された教示及び指示を利用したいくつかのフィルタリングアプリケーションのための高速化された推定値を与える。

【００６５】

【表１】

処理	スピードアップ
9−7ウェーブレット	1.7
バイト係数による3×3フィルタ	4.3
透かし補正	6.8

他の実施例

ＳＩＭＤレジスタを利用したコンテンツデータの効率的なフィルタリング及び畳み込み処理を提供するコンピュータアーキテクチャの一実施例のいくつかの特徴が説明されてきた。しかしながら、コンピュータアーキテクチャの様々な実施形態により、上述の特徴の補完、補助及び／あるいは置換を含む多くの特徴が提供される。これらの特徴は、コンピュータアーキテクチャの一部、あるいは異なる実施形態でのソフトウェアまたはハードウ

ェア要素の一部として実現される。また、上記記述では説明のため、本発明の完全な理解を与えるため特定の用語が利用された。しかしながら、本発明の実践のために特定の詳細が必ずしも必要でないということは当業者には明らかであろう。

【００６６】

さらに、ここで説明された実施例は、ＳＩＭＤレジスタを使ったコンテンツデータの効率的なフィルタリング及び畳み込み処理のためのシステムを対象としているが、本発明の教示は他のシステムに適用可能であるということは当業者には理解されるであろう。実際、画像、音声及び映像データの処理システムは、本発明の範囲及び趣旨から逸脱することなく本発明の教示の範囲内に入る。上述の実施例は、本発明の原理及びその実践的適用を最も良く説明するため選ばれ、記述された。これらの実施例は、当業者が特定の利用形態に適するよう様々な修正を行うことにより、発明及び様々な実施例を最も良く利用できるよう選ばれた。

【００６７】

本発明の実施例は、従来技術に対する多くの効果を提供する。本発明は、複数の配列長、データサイズ及び係数符号に対するフィルタリング／畳み込み処理を効率的に実行することができる。これらの処理は、小さな単一命令多重データ（ＳＩＭＤ）命令群の一部であるいくつかの命令を利用することにより実行される。従って、本発明は不要なデータタイプ変更を回避する。この結果、不要なデータタイプ変更を回避することによって、本発明は命令あたりのＳＩＭＤ処理数を最大化する一方、乗加算演算のような算術処理のためデータを順序付ける必要があるクロックサイクル数を大きく減少させることができる。

【００６８】

図１５は、本発明による右方向平行シフトマージ処理を実行する論理回路を含む一例となるプロセッサのマイクロアーキテクチャのブロック図である。右方向シフトマージ処理はまた、上記説明と同様に、レジスタマージ処理及びレジスタマージ命令として呼ばれる。右方向シフトマージ命令（ＰＳＲＭＲＧ）の一実施例のため、当該命令は図１、９Ａ及び９Ｂのレジスタマージ処理１６７と同じ結果をもたらす。イン・オーダーフロントエンド（ｉｎ－ｏｒｄｅｒ　ｆｒｏｎｔ　ｅｎｄ）１００１は、実行対象のマクロ命令を取り込み、プロセッサパイプラインでの後の利用のためそれらを用意するプロセッサ１０００の一部である。本実施例のフロントエンドは複数のユニットを含んでいる。命令プレフェッチャ（ｉｎｓｔｒｕｃｔｉｏｎ　ｐｒｅｆｅｔｃｈｅｒ）１０２６は、メモリからマクロ命令を取り込み、それらを命令デコーダ１０２６に供給する。そして命令デコーダ１０２８は、このマクロ命令をマシーンが実行可能なマイクロ命令あるいはマイクロ処理（またはマイクロｏｐまたはｕｏｐと呼ばれる）と呼ばれる要素に復号する。トレースキャッシュ１０３０は、復号化されたｕｏｐを受け取り、実行のためｕｏｐキュー１０３４においてそれらを順序付けされたプログラムシーケンスあるいはトレースに分解する。トレースキャッシュ１０３０が複雑なマクロ命令に直面すると、マイクロコードＲＯＭ１０３２が当該処理の完了に必要なｕｏｐを提供する。

【００６９】

多くのマクロ命令が１つのマイクロｏｐに変換される一方、他のマクロ命令は完全な処理の完了のため複数のマイクロｏｐを必要とするかもしれない。本実施例では、マクロ命令の完了のため４より多くのマイクロｏｐが必要な場合、デコーダ１０２８はマイクロコードＲＯＭ１０３２にアクセスし、マクロ命令を実行する。一実施例では、右方向平行シフトマージアルゴリズムのための命令は、多数のマイクロｏｐが処理の実行に必要である場合、マイクロコードＲＯＭ１０３２に格納することができる。トレースキャッシュ１０３０は、入力ポイントＰＬＡ（Ｐｒｏｇｒａｍｍａｂｌｅ　Ｌｏｇｉｃ　Ａｒｒａｙ）を参照し、マイクロコードＲＯＭ１０３２における分割アルゴリズムのためのマイクロコードシーケンスを読み込むための正しいマイクロ命令ポインタを決定する。マイクロコードＲＯＭ１０３２が現在のマクロ命令に対するマイクロｏｐの順序付けを完了すると、マシーンのフロントエンド１００１はトレースキャッシュ１０３０からマイクロｏｐの取り込みを再開する。

【００７０】
いくつかのＳＩＭＤ及び他のマルチメディアタイプの命令は複雑な命令とみなされる。浮動小数点に関する大部分の命令もまた複雑な命令である。さらに、命令デコーダ１０２８が複雑なマクロ命令に直面すると、マイクロコードＲＯＭ１０３２は当該マクロ命令のためのマイクロコードシーケンスを抽出するために、適当な位置でアクセスされる。このマクロ命令の実行に要する様々なマイクロｏｐが、適当な整数及び浮動小数点実行ユニットにおける実行のため、アウト・オブ・オーダー実行エンジン１００３に通信される。
【００７１】
アウト・オブ・オーダー実行エンジン１００３では、実行のためにマイクロ命令が用意されている。アウト・オブ・オーダー実行論理は、マイクロ命令がパイプラインに入り、実行のためスケジューリングされるとき、パフォーマンスを最適化するためマイクロ命令のフローを平滑化及び順序調整をするための複数のバッファを有する。割り当て論理は、各ｕｏｐが実行に必要とするマシーンバッファやリソースを割り当てる。レジスタリネーム論理は、論理レジスタをレジスタファイルの入力に改名する。割り当て論理はまた、メモリスケジューラ、高速スケジューラ１００２、低速／通常浮動小数点スケジューラ１００４、及びシンプル浮動小数点スケジューラ１００６の命令スケジューラの前に、メモリ処理及び非メモリ処理のための２つのｕｏｐキューの１つにおける各ｕｏｐに対する入力を割り当てる。ｕｏｐスケジューラ１００２、１００４及び１００６は、スケジューラの従属入力レジスタオペランドソースの準備状況と、ｕｏｐが処理の遂行に必要とする実行リソースの利用可能状況に基づき、ｕｏｐの実行準備がいつ整うかを判断する。本実施例の高速スケジューラ１００２がメインクロックサイクルの半サイクルごとにスケジューリングを行う一方、その他のスケジューラはメインプロセッサクロックサイクルあたり１回だけスケジューリングを行うことができる。スケジューラはディスパッチポートを調停して、実行のためのｕｏｐをスケジューリングする。
【００７２】
レジスタファイル１００８と１０１０は、スケジューラ１００２、１００４及び１００６と、実行ブロック１０１１の実行ユニット１０１２、１０１４、１０１６、１０１８、１０２０、１０２２及び１０２４との間に配置される。整数及び浮動小数点演算のためにそれぞれレジスタファイル１００８と１０１０がある。本実施例の各レジスタファイル１００８と１０１０はまた、まだレジスタファイルに書き込まれていない完了結果を新しい従属ｕｏｐにバイパスあるいは転送するバイパスネットワークを含む。整数レジスタファイル１００８と浮動小数点レジスタファイル１０１０はまた、互いにデータの通信を行うことができる。一実施例において、整数レジスタファイル１００８は２つのレジスタファイルに分割され、その一方は下位３２ビットデータ用のレジスタファイルであり、もう一方は上位３２ビットデータ用のレジスタファイルである。一実施例の浮動小数点レジスタファイルは１０１０は、１２８ビット幅の入力を有する。これは浮動小数点命令は典型的に、６４から１２８ビット幅のオペランドを有するからである。
【００７３】
実行ブロック１０１１は、命令を実際に実行する実行ユニット１０１２、１０１４、１０１６、１０１８、１０２０、１０２２及び１０２４を含む。この部分は、マイクロ命令が実行に必要とする整数及び浮動小数点データオペランド値を格納するレジスタファイル１００８と１０１０を含む。本実施例のプロセッサ１０００は、アドレス生成ユニット（ＡＧＵ）１０１２、ＡＧＵ１０１４、高速ＡＬＵ１０１６、高速ＡＬＵ１０１８、低速ＡＬＵ１０２０、浮動小数点ＡＬＵ１０２２及び浮動小数点移動ユニット１０２４からなる複数の実行ユニットから構成される。本実施例において、浮動小数点実行ブロック１０２２と１０２４は、浮動小数点ＭＭＸ、ＳＩＭＤ及びＳＳＥ処理を実行する。本実施例の浮動小数点ＡＬＵ３２２は、割算、平方根及び剰余に関するマイクロｏｐを実行するための６４ビット単位浮動小数点割算器を有する。本発明の実施例では、浮動小数に関する任意の処理は浮動小数点ハードウェアにより引き起こされる。例えば、整数形式と浮動小数形式間の変換には、浮動小数点レジスタファイルが関係する。同じように、浮動小数割算処

理は浮動小数点割算器において引き起こされる。他方、非浮動小数点数及び整数タイプは整数ハードウェアリソースにより処理される。単純かつ頻繁に使用されるＡＬＵ演算は、高速ＡＬＵ実行ユニット１０１６と１０１８において処理される。本実施例の高速ＡＬＵ１０１６と１０１８は、半分のクロックサイクルの効果的な待ち時間により高速処理を実行することができる。一実施例では、大部分の複雑な整数演算は低速ＡＬＵ１０２０に渡される。低速ＡＬＵ１０２０は、乗算、シフト、フラグ論理及び分岐処理のような長い待ち時間を要するタイプの処理用の整数実行ハードウェアを含む。メモリロード／ストア処理は、ＡＧＵ１０１２と１０１４により実行される。本実施例では、整数ＡＬＵ１０１６、１０１８及び１０２０は、６４ビットデータオペランドに対する整数処理の実行に関して説明される。他の実施例では、ＡＬＵ１０１６、１０１８及び１０２０は、１６、３２、１２８、２５６などの様々なデータビットをサポートするよう実現することができる。同じように、浮動小数点ユニット１０２２と１０２４は、様々なビット幅を有するオペランドをサポートするよう実現することができる。一実施例では、浮動小数点ユニット１０２２と１０２４は、ＳＩＭＤ及びマルチメディア命令に関して、１２８ビット幅のＰａｃｋｅｄデータオペランドにおいて実行される。

【００７４】

本実施例では、ｕｏｐスケジューラ１００２、１００４及び１００６は、親ロードが実行を完了する前に、従属処理をディスパッチする。ｕｏｐがプロセッサ１０００において投機的にスケジューリング及び実行されるとき、プロセッサ１０００はまたメモリミスを処理するための論理を含む。データキャッシュにおいてデータロードがミスする場合、一時的に誤ったデータをスケジューラに残したパイプラインにおけるフライト（ｆｌｉｇｈｔ）での従属処理が存在しうる。リプレイ機構が、誤ったデータを利用する命令を追跡及び再実行する。従属処理のみがリプレイされる必要があり、独立した処理は完了させることができる。プロセッサの一実施例のスケジューラ及びリプレイ機構はまた、拡張された精度整数割算処理の命令シーケンスを獲得するよう構成されている。

【００７５】

「レジスタ」という単語は、オペランドを特定するマクロ命令の一部として利用されるオン・ボードプロセッサ記憶領域を参照するのにここでは使われる。言い換えると、ここで呼ばれるレジスタとは、プロセッサ外部から（プログラマーの視点から）見ることができるものである。しかしながら、ここで述べられるレジスタは、例えば、専用物理レジスタ、レジスタリネーミングを利用した動的に割り当てられた物理レジスタ、専用物理レジスタと動的に割り当てられる物理レジスタとを組み合わせたものなどのような様々なテクニックを利用したプロセッサ内部の回路により実現することができる。

【００７６】

以下の説明のため、レジスタは、カリフォルニア州サンタクララのインテルコーポレーションからのＭＭＸ技術が可能なマイクロプロセッサにおける６４ビット幅ＭＭＸ（商標）レジスタ（ｍｍレジスタ）のようなＰａｃｋｅｄデータの保持が可能なデータレジスタであると解釈される。このようなＭＭＸレジスタは、整数及び浮動小数点の両方の形式で利用可能であり、ＳＩＭＤとＳＳＥ命令を伴うＰａｃｋｅｄデータ要素により動作可能である。同様に、ＳＳＥ２技術に関する１２８ビット幅ＸＭＭレジスタもまた、そのようなＰａｃｋｅｄデータオペランドの保持に利用可能である。

【００７７】

以下の図面による例では、多数のデータオペランドが説明される。簡単化のため、データセグメントはＡからアルファベット順にラベル付けされ、Ａは最下位アドレスに、Ｚは最上位アドレスに位置する。従って、Ａはアドレス０に、Ｂはアドレス１に、Ｃはアドレス３に、以下同様にしてラベル付けされる。いくつかの例では、データシーケンスは逆アルファベット順に配置された文字により表されるが、アドレッシングは依然として０にあるＡから、１にあるＢという順で開始される。概念的には、一実施例のための右方向シフトマージと同様に、右方向シフト処理は、シーケンスがＤ、Ｃ、Ｂ、Ａである場合、最下位アドレスのデータセグメントを右方向にシフトすることに関する。右方向のシフトは、

１つのデータブロック分のデータ要素を固定線に沿って単に右方向にシフトする。さらに、概念的には右方向シフトマージ処理は、最右データセグメントを１つのオペランドから他のデータオペランドの左側に、あたかもこれら２つのオペランドが連続するように、右方向にシフトする。

【００７８】

図１６Ａは、本発明によるデータオペランドに対する右方向平行シフトマージ処理を実行する論理の一実施例のブロック図である。本実施例の右方向シフトマージ（またはレジスタシフト）処理のための命令（ＰＳＲＭＲＧ）は、第１データオペランド１１０２、第２データオペランド１１０４及びシフトカウント１１０６からなる３つの情報から開始される。一実施例では、シフトＰＳＲＭＲＧ命令が１つのマイクロ処理に復号される。他の実施例では、当該命令はデータオペランドに対するシフトマージ処理を実行する様々なマイクロｏｐに復号されてもよい。本実施例では、データオペランド１１０２と１１０４はレジスタ／メモリに格納された６４ビット幅のデータ片であり、シフトカウント１１０６は８ビット幅の即値である。特定の実現に応じて、データオペランドとシフトカウントは、それぞれ１２８／２５６ビットと１６ビットのような他の幅であってもよい。本実施例での第１オペランド１１０２は、Ｐ、Ｏ、Ｎ、Ｍ、Ｌ、Ｋ、Ｊ及びＩの８つのデータセグメントから構成される。第２オペランドもまた、Ｈ、Ｇ、Ｆ、Ｅ、Ｄ、Ｃ、Ｂ及びＡの８つのデータセグメントから構成される。これらデータセグメントは等しい長さを有し、それぞれ１バイト（８ビット）データから構成される。しかしながら、本発明の他の実施例はより長い１２８ビットのオペランドにより動作する。この場合、各データセグメントは１バイト（８ビット）からなり、１２８ビット幅のオペランドは１６バイト幅のデータセグメントを有する。同様に、各データセグメントがダブルワード（３２ビット）またはクアドワード（６４ビット）である場合、１２８ビットオペランドはそれぞれ４ダブルワード幅あるいは２クアドワード幅のデータセグメントを有する。従って、本発明の実施例は特定の長さのデータオペランド、データセグメントあるいはシフトカウントに限定されるものでなく、各実現形態に対し適当なサイズとすることができる。

【００７９】

オペランド１１０２と１１０４は、レジスタ、メモリ領域、レジスタファイルあるいはそれらを組み合わせたものに格納することができる。データオペランド１１０２と１１０４とカウント１１０６が、右方向シフトマージ命令と共にプロセッサ内の実行ユニット１１１０に送信される。右方向シフトマージ命令が実行ユニット１１１０に届くまでに、当該命令はプロセッサパイプラインにおいて復号されるべきであった。従って、右方向シフトマージ命令は、マイクロ処理（ｕｏｐ）による形式あるいは他の復号形式でありうる。本実施例では、２つのデータオペランド１１０２と１１０４が連結論理と仮のレジスタにおいて受信される。連結論理は、これら２つのオペランドに対しデータセグメントをマージ／ジョインし、新しいデータブロックを仮のレジスタに配置する。ここで、この新しいデータブロックは、Ｐ、Ｏ、Ｎ、Ｍ、Ｌ、Ｋ、Ｊ、Ｉ、Ｈ、Ｇ、Ｆ、Ｅ、Ｄ、Ｃ、Ｂ及びＡの１６のデータセグメントから構成される。本実施例は６４ビット幅のオペランドにより動作するので、合成されたデータを保持する必要のある仮のレジスタは１２８ビット幅である。１２８ビット幅のデータオペランドに対しては、２５６ビット幅の仮レジスタが必要とされる。

【００８０】

実行ユニット１１１０における右方向シフト論理１１１４は、仮レジスタのコンテンツを取得し、カウント１１０６により要求に従い、ｎデータセグメントだけデータブロックを右方向に論理シフトする。本実施例では、カウント１１０６は右方向にシフトするバイト数を示す。特定の実施形態に応じて、カウント１１０６を使ってデータセグメントの粒度に応じたシフトされるべきビット数、ニブル（ｎｉｂｂｌｅ）数、ワード数、ダブルワード数、クアドワード数などを示すことができる。本実施例では、仮レジスタのコンテンツが３バイトだけシフトされるようｎは３に等しくされる。各データセグメントがワードあるいはダブルワード幅であるとき、カウントはそれぞれシフトされるべきワード数ある

いはダブルワード数を表す。本実施例では、レジスタのデータが右方向にシフトされるとき、空のスペースを埋めるために、仮レジスタの左側から０がシフト入力される。従って、シフトカウント１１０６がデータオペランドのデータセグメント数（この場合、８）より大きい場合、１つ以上の０が結果となる１１０８に現れる。さらに、シフトカウント１１０６が両方のオペランドに対しデータセグメントの総数以上であれば、すべてのデータセグメントがシフトにより破棄されるので、結果はすべて０となる。右方向シフト論理１１１４は、仮レジスタから適当な個数のデータセグメントを結果１１０８として出力する。他の実施例では、出力マルチプレクサあるいはラッチを、結果を出力する右方向シフト論理の後に備えることができる。本実施例では、当該結果は６４ビット幅であり、８バイトを有する。２つのデータオペランド１１０２と１１０４に対する右方向シフトマージ処理により、結果はＫ、Ｊ、Ｉ、Ｈ、Ｇ、Ｆ、Ｅ及びＤの８つのデータセグメントから構成される。

【００８１】

図１６Ｂは、右方向シフトマージ処理を実行する論理の他の実施例のブロック図である。図１６Ａの前記実施例と同様に、本実施例の右方向シフトマージ処理は、６４ビット幅の第１データオペランド１１０２、６４ビット幅の第２データオペランド１１０４及び８ビット幅のシフトカウント１１０６の３つの情報から開始される。シフトカウント１１０６は、データセグメントのシフト数を示す。本実施例では、カウント１１０６はバイト数に関し記述される。他の実施例では、カウントはデータのシフトされるべきビット数、ニブル数、ワード数、ダブルワード数、あるいはクアドワード数を示してもよい。本実施例における第１及び第２オペランドはそれぞれ、８に等しい長さのバイトサイズデータセグメントからなり、第１オペランド１１０２は８つのデータセグメント（Ｈ、Ｇ、Ｆ、Ｅ、Ｄ、Ｃ、Ｂ、Ａ）、第２オペランド１１０４は８つのデータセグメント（Ｐ、Ｏ、Ｎ、Ｍ、Ｌ、Ｋ、Ｊ、Ｉ）から構成される。カウントｎは３に等しい。本発明の他の実施例は、例えば、１２８／２５６／５１２ビット幅のオペランド、ビット／バイト／ワード／ダブルワード／クアドワードのサイズのデータセグメント、８／１６／３２ビット幅のシフトカウントのような他の長さのオペランド及びデータセグメントにより動作することも可能である。従って、本発明の実施例は、特定の長さのデータオペランド、データセグメントあるいはシフトカウントに限定されず、各実施形態に適したサイズとすることができる。

【００８２】

データオペランド１１０２と１１０４、及びカウント１１０６は、右方向シフトマージ命令と共にプロセッサ内の実行ユニット１１２０に送信される。本実施例では、第１データオペランド１１０２と第２データオペランド１１０４が、それぞれ左方向シフト論理１１２２と右方向シフト論理１１２４において受信される。カウント１１０６もまた、シフト論理１１２２と１１２４に送られる。左方向シフト論理１１２２は、第１オペランド１１０２のデータセグメントを「第１オペランドのデータセグメント数－ｎ」個のセグメントだけ左方向にシフトする。データセグメントが左にシフトされると、空いたスペースを埋めるように左側から０がシフト入力される。この場合、８つのデータセグメントがあるので、第１オペランド１１０２は８－３＝５だけ左にシフトされる。第１オペランド１１０２は、論理ＯＲゲート１１２６においてマージのため正しくデータを配置するために、この異なる値だけシフトされる。ここでの左方向へのシフト後、第１データオペランドは、Ｋ、Ｊ、Ｉ、０、０、０、０、０となる。カウント１１０６がオペランドのデータセグメント数より大きい場合、左方向へのシフト計算は負の値となり、これは左方向への負のシフトを表す。負のカウントのよる左方向シフト論理は、負の方向へのシフトとして解釈され、実質的に右方向へのシフト論理となる。左方向への負のシフトにより、第１オペランド１１０２の左側から０となる。

【００８３】

同じように、右方向シフト論理１１２４は、第２オペランドのデータセグメントをｎセグメントだけ右方向にシフトする。データセグメントが右にシフトされると、殻のスペースを埋めるために左側から０がシフト入力される。第２データオペランドは、０、０、０

、Ｈ、Ｇ、Ｆ、Ｅ、Ｄとなる。シフトされたオペランドは左シフト論理１１２２または右シフト論理１１２４から出力され、論理ＯＲゲート１１２６においてマージされる。ＯＲゲートは、データセグメントの論理ＯＲ処理を実行し、本実施例の６４ビット幅の結果１１０８を与える。「Ｋ、Ｊ、Ｉ、０、０、０、０、０」と「０、０、０、Ｈ、Ｇ、Ｆ、Ｅ、Ｄ」のＯＲ処理により、８バイトのＫ、Ｊ、Ｉ、Ｈ、Ｇ、Ｆ、Ｅ、Ｄから構成される結果１１０８が生成される。この結果は、図１６Ａにおける本発明の第１実施例のものと同じである。ここで、オペランドのデータセグメント数より大きなカウントｎ１１０６に対しては、適当な数の０が結果の左側から現れる。さらに、カウント１１０６が両方のオペランドのデータ要素数の合計以上である場合、当該結果はすべて０より構成される。

【００８４】

図１７Ａは、本発明の第１実施例による右方向平行シフトマージ命令の動作を示す。この説明において、ＭＭ１１２０４、ＭＭ２１２０６、ＴＥＭＰ１２３２及びＤＥＳＴ１２４２は一般に、オペランドまたはデータブロックと呼ばれるが、それに限定されるものでなく、レジスタ、レジスタファイル及びメモリ領域を含む。一実施例では、ＭＭ１１２０４とＭＭ２１２０６は、６４ビット幅のＭＭＸレジスタ（あるいはいくつかの例では「ｍｍ」と呼ばれる）である。状態Ｉ１２００において、シフトカウントｉｍｍ［ｙ］１２０２、第１オペランドＭＭ１［ｘ］１２０４及び第２オペランドＭＭ２［ｘ］１２０６が、右方向平行シフトマージ命令と共に送られる。カウント１２０２は、ｙビット幅の即値である。第１オペランド１２０４と第２オペランド１２０６はそれぞれ、各データセグメントが１バイト（８ビット）の場合、ｘデータセグメントを含み、８ｘビットの合計幅を有するデータブロックである。第１オペランドと第２オペランドはそれぞれ、多数のより小さいデータセグメントによりパックされる。本実施例では、第１データオペランドＭＭ１１２０４は、Ｐ１２１１、Ｏ１２１２、Ｎ１２１３、Ｍ１２１４、Ｌ１２１５、Ｋ１２１６、Ｊ１２１７、Ｉ１２１８の８に等しい長さのデータセグメントから構成される。同じように、第２データオペランドＭＭ２１２０６は、Ｈ１２２１、Ｇ１２２２、Ｆ１２２３、Ｅ１２２４、Ｄ１２２５、Ｃ１２２６、Ｂ１２２７、Ａ１２２８の８に等しい長さのデータセグメントから構成される。従って、これらのデータセグメントのそれぞれは「ｘ・８」ビット幅となる。これより、ｘが８のとき、各オペランドは８バイトあるいは６４ビット幅となる。他の実施例では、データ要素は、ニブル（４ビット）、ワード（１６ビット）、ダブルワード（３２ビット）、クアドワード（６４ビット）などであってもよい。他の実施例では、ｘは１６、３２、６４のデータ要素幅であってもよい。本実施例では、カウントｙは８に等しく、即値はバイトで表される。他の実施例では、ｙは４、１６、３２などのビット幅であってもよい。さらに、カウント１２０２は、即値に限定されず、レジスタまたはメモリ領域に格納される。

【００８５】

オペランドＭＭ１１２０４とＭＭ２１２０６は、状態ＩＩ１２０３においてマージされ、２ｘデータ要素幅（あるいはこの場合にはバイト）の仮のデータブロックＴＥＭＰ［２ｘ］１２３２が生成される。本実施例のマージされたデータ１２３２は、Ｐ、Ｏ、Ｎ、Ｍ、Ｌ、Ｋ、Ｊ、Ｉ、Ｈ、Ｇ、Ｆ、Ｅ、Ｄ、Ｃ、Ｂ及びＡとして配置される１６のデータセグメントから構成される。８バイト幅の窓１２３４は、最右端から始まる仮のデータブロック１２３２の８つのデータセグメントから構成される。従って、窓１２３４の右端は、窓１２３４がＨ、Ｇ、Ｆ、Ｅ、Ｄ、Ｃ、Ｂ及びＡのデータセグメントから構成されるようデータブロック１２３２の右端から並べられる。シフトカウントｎ１２０２は、マージされたデータを右方向にシフトする所望のシフト数を示す。このカウント値は、ビット、ニブル、バイト、ワード、ダブルワード、クアドワードなど、あるいは特定のデータセグメント数に関するシフト数を示すよう実現されてもよい。カウント値１２０２に基づき、データブロック１２３２はｎデータセグメントだけ右方向へシフトされる（１２３６）。本実施例では、ｎは３に等しく、データブロックは３だけ右にシフトされる。これのもう１つの見方は、窓１２３４を反対方向へシフトするというものである。言い換えると、窓１

２３４は概念的には、仮のデータブロック１２３２の右端から左に３つシフトしたとみなすことができる。一実施例において、シフトカウントｎが合成されたデータブロックにおいて与えられるデータセグメントの総数２ｘより大きい場合、結果はすべて０から構成されるであろう。同じように、シフトカウントｎが第１オペランド１２０４におけるデータセグメント数ｘ以上である場合、結果はそれの右端から１つ以上の０を含むことになる。状態ＩＩＩ１２４０において、窓１２３４により構成されるデータセグメント「Ｋ、Ｊ、Ｉ、Ｈ、Ｇ、Ｆ、Ｅ、Ｄ」が、結果としてｘデータ要素幅目的ＤＥＳＴ［ｘ］１２４２に出力される。

【００８６】

図１７Ｂは、第２実施例による右方向シフトマージ命令の動作を示す。状態Ｉ１２５０において、右方向シフトマージ命令は、ｙビットのカウントｉｍｍ［ｙ］、ｘデータセグメントの第１データオペランドＭＭ１［ｘ］、及びｘデータセグメントの第２データオペランドＭＭ２［ｘ］により伴われる。図１７Ａの実施例に関し、ｙは８に等しく、ｘも８に等しい。ここでＭＭ１とＭＭ２はそれぞれ６４ビットまたは８バイト幅である。本実施例の第１及び第２オペランドはそれぞれ同じサイズのデータセグメント数によりパックされる。この場合、各オペランドは１バイト幅であり、第１オペランド１２０４は「Ｐ１２１１、Ｏ１２１２、Ｎ１２１３、Ｍ１２１４、Ｌ１２１５、Ｋ１２１６、Ｊ１２１７、Ｉ１２１８」、第２オペランド１２０６は「Ｈ１２２１、Ｇ１２２２、Ｆ１２２３、Ｅ１２２４、Ｄ１２２５、Ｃ１２２６、Ｂ１２２７、Ａ１２２８」から構成される。

【００８７】

状態ＩＩ１２６０において、第１オペランド１２０４と第２オペランド１２０６をシフトするのにシフトカウントｎ１２０２が利用される。本実施例のカウントは、マージされたデータを右にシフトするデータセグメント数を示す。本実施例では、第１オペランド１２０４と第２オペランド１２０６とのマージ処理前に、シフト処理は行われる。この結果、第１オペランド１２０４は異なってシフトされる。本実施例では、第１オペランド１２０４は（ｘ－ｎ）データセグメント数だけ左にシフトされる。この「ｘ－ｎ」の計算により、後のデータマージ処理での適切なデータ配置が可能になる。従って、カウントｎが３のとき、第１オペランド１２０４は５データセグメントあるいは５バイトだけ左方向にシフトされる。空のスペースを埋めるために、左端から０がシフト入力される。しかしながら、シフトカウントｎ１２０２が第１オペランド１２０４において利用可能なデータセグメント数ｘより大きい場合、「ｘ－ｎ」の左方向シフト計算は、実質的に負の左方向へのシフトを示す負の値を生じる。一実施例では、負のカウントによる左方向への論理シフトは負の方向への左シフトとして解釈され、実質的に右方向への論理シフトとなる。負の左方向へのシフトにより、第１オペランド１２０４の左側から０が配置される。同じように、第２オペランド１２０６は、３のシフトカウントだけ右にシフトされ、空白を埋めるため左方向から０がシフト入力される。第１オペランド１２０４と第２オペランド１２０６に行われたシフト結果はそれぞれ、ｘデータセグメント幅のレジスタＴＥＭＰ１１２６６とＴＥＭＰ２１２６８に格納される。ＴＥＭＰ１１２６６とＴＥＭＰ２１２６８からのシフト結果がマージされ（１２７２）、状態ＩＩＩ１２７０においてレジスタＤＥＳＴ１２４２で所望のシフトマージデータが生成される。シフトカウントｎ１２０２がｘより大きい場合、結果の中に左端から１つ以上の０を含む。さらに、シフトカウントｎ１２０２が２ｘ以上である場合、ＤＥＳＴ１２４２における結果はすべて０から構成されている。

【００８８】

上記実施例において、図１７Ａ及び図１７Ｂに示されるように、ＭＭ１とＭＭ２の一方または両方がＭＭＸ／ＳＳＥ技術により可能なプロセッサの６４ビットデータレジスタ、あるいはＳＳＥ２技術による１２８ビットデータレジスタであってもよい。実施形態に応じて、これらのレジスタは６４／１２８／２５６ビット幅であってもよい。同じように、ＭＭ１とＭＭ２の一方または両方がレジスタ以外のメモリ領域であってもよい。一実施例のプロセッサアーキテクチャでは、ＭＭ１とＭＭ２は上述のような右方向シフトマージ命令（ＰＳＲＭＲＧ）に対するソースオペランドである。シフトカウントＩＭＭはまた、こ

のようなＰＳＲＭＲＧ命令の即値である。一実施例では、結果のための目的ＤＥＳＴはまた、ＭＭＸまたはＸＭＭデータレジスタである。さらに、ＤＥＳＴはソースオペランドの１つと同じレジスタであってもよい。例えば、あるアーキテクチャでは、ＰＳＲＭＲＧ命令は、第１ソースオペランドＭＭ１と第２ソースオペランドＭＭ２を有する。結果に対する所定の目的は、この場合ＭＭ！である第１ソースオペランドのレジスタでありうる。

【００８９】

図１８Ａは、データオペランドを並列に右方向シフト及びマージする方法の一実施例を示すフローチャートを示す。Ｌの長さが一般に、オペランドとデータブロックの幅を表すのに使われる。特定の実施例に応じて、Ｌはデータセグメント数、ビット数、倍と数、ワード数などに関して幅を表すのに使われてもよい。ブロック１３０２において、長さＬの第１データオペランドが、シフトマージ処理の実行において利用するため受信される。このシフトマージ処理のための長さＬの第２データオペランドがまたブロック１３０４において受信される。ブロック１３０６において、ビット／ニブル／バイト／ワード／ダブルワード／クアドワードでのデータセグメント数あるいは距離を示すシフトカウントが受信される。ブロック１３０８における実行論理は、第１オペランドと第２オペランドを連結する。一実施例において、長さ２Ｌの仮のレジスタにおいて、連結されたデータブロックが保持される。他の実施例では、マージされたデータはメモリ領域に保持される。ブロック１３１０において、連結されたデータブロックはシフトカウントだけ右にシフトされる。このカウントがデータセグメントカウントとして表されている場合、データブロックはそのデータブロックだけ右にシフトされ、空白を埋めるためデータブロックの最上位エンド（ｍｏｓｔ　ｓｉｇｎｉｆｉｃａｎｔ　ｅｎｄ）に沿って左から０がシフト入力される。例えば、このカウントがビットあるいはバイトで表されていれば、データブロックはその距離だけ同じように右方向にシフトされる。ブロック１３１２において、長さＬの結果がシフトされたデータブロックの左側あるいは最下位エンド（ｌｅａｓｔ　ｓｉｇｎｉｆｉｃａｎｔ　ｅｎｄ）から生成される。一実施例において、長さＬのデータセグメントが、シフトされたデータブロックから目的レジスタあるいはメモリ領域にマックス（ｍｕｘ）される。

【００９０】

図１８Ｂは、データの右方向へのシフト及びマージ方法の他の実施例を示すフロー図である。ブロック１３５２において、長さＬの第１データオペランドが右方向シフト及びマージ処理のため受信される。長さＬの第２データセグメントがブロック１３５４において受信される。ブロック１３５６において、所望の右方向シフト距離を示すシフトカウントが受信される。ブロック１３５８において、シフトカウントによる計算に基づき、第１データオペランドが左方向にシフトされる。一実施例の計算は、シフトカウントをＬから差し引くことからなる。例えば、オペランドの長さＬとシフトカウントがデータセグメントに関するものである場合、第１オペランドは「Ｌ－シフトカウント」セグメントだけ左にシフトされ、当該オペランドの最下位エンドから０がシフト入力される。同じように、Ｌがビットにより、カウントがバイトにより表される場合、第１オペランドは「Ｌ－シフトカウント・８」ビットだけ左にシフトされる。ブロック１３６０において、第２データオペランドはシフトカウント分右にシフトされ、空白を埋めるため第２オペランドの最上位エンドから０がシフト入力される。ブロック１３６２において、シフトされた第１オペランドとシフトされた第２オペランドはマージされ、長さＬの結果が生成される。一実施例では、このマージ処理により、第１及び第２オペランドの両方からの所望のデータセグメントからなる結果が生成される。

【００９１】

コンピュータにおいてますますよく使われる利用方法は、サイズの大きな映像及び音声ファイルの操作に関するものである。これらの映像及び音声は典型的には広帯域幅のネットワークあるいは大容量記憶媒体を介し転送されるとしても、トラフィック処理のため依然としてデータ圧縮は必要である。この結果、様々な圧縮アルゴリズムが、多くの一般的な音声、画像及び映像フォーマットのための表現または符号化スキームの重要部分になり

つつある。ＭＰＥＧ（Ｍｏｔｉｏｎ　Ｐｉｃｔｕｒｅ　Ｅｘｐｅｒｔ　Ｇｒｏｕｐ）規格の１つに従う映像は、圧縮を利用した１つの適用例である。ＭＰＥＧ映像は、レイヤ階層に分解され、エラー処理、ランダム探索と編集、及び同期に利用される。

【００９２】

例示のため、１つのＭＰＥＧ映像を構成するこれらのレイヤが簡潔に説明される。最上位レベルには、自己完結した（ｓｅｌｆ－ｃｏｎｔａｉｎｅｄ）ビットストリームを含む映像シーケンスレイヤがある。第２レイヤには、１つ以上のイントラ及び／または非イントラフレーム群からなる画像群がある。第３レイヤには画像レイヤがあり、その次のレイヤはスライスレイヤ（ｓｌｉｃｅ　ｌａｙｅｒ）である。各スライスは、以下に限定されるものではないが、最もよく利用されている典型的な映像アプリケーションでの行ベースに基づくラスター順序のマクロブロックの連続したシーケンスである。各スライスは、１６×１６の輝度画素（ｌｕｍｉｎａｎｃｅ　ｐｉｘｅｌ）配列と、それに対応する２つの８×８の色光度画素（ｃｈｒｏｍｉｎａｎｃｅ　ｐｉｘｅｌ）配列とを有するマクロブロックまたは画像データ要素から構成される。マクロブロックは、変換符号化のようなさらなる処理のため相異なる複数の８×８ブロックに分割される。マクロブロックは、動き補償及び動き予測のための基本ユニットであり、それに関連付けされた動きベクトルを有する。実施例に応じて、マクロブロックは１６行１６列であってもよいし、あるいは様々なサイズであってもよい。

【００９３】

ＭＰＥＧ映像で使われる時間予測テクニックは動き予測に基づいている。動き予測は、連続する映像フレームは一般に、これらフレーム内で動いているオブジェクトに起因する変化以外は、同じであるという前提に基づくものである。フレーム間の動きがゼロである場合、エンコーダは現在フレームを過去あるいは予測フレームの複製として容易かつ効率的に予測することができる。前フレーム（ｐｒｅｖｉｏｕｓ）はまた、参照フレームと呼ばれる。他の実施例では、参照フレームは、当該シーケンスにおける次のフレームあるいは他のフレームであるかもしれない。動き予測の実施例は、前フレームと現在フレームを比較する必要はない。他の任意のフレームがこの比較において利用できる。エンコーダに送信される必要のある情報は、もとの参照フレームから画像を復元するのに必要な構文オーバヘッド（ｓｙｎｔａｃｔｉｃ　ｏｖｅｒｈｅａｄ）となる。しかしながら、画像間に動きが存在する場合、状況はより複雑である。最もよく一致するマクロブロックと現在マクロブロックとの差は、理想的には多くが０となることである。マクロブロックを符号化するとき、最もよく一致するマクロブロックと現在マクロブロックとの差が変換及び量子化される。一実施例では、量子化された値が、圧縮のため可変長符号化処理に通信される。０は良好に圧縮することができるので、差が０である値を多く有する最もよく一致するマクロブロックが望ましい。動きベクトルはまた、これらの差の値から導出することもできる。

【００９４】

図１９Ａは、動き予測の第１実施例を示す。左のフレーム１４０２は、棒線画と案内標識を含む前映像フレームのサンプルである。右のフレーム１４０４は、同様の棒線画と案内標識を含む現在映像フレームのサンプルである。現在フレーム１４０４では、パン撮りにより、案内標識が前フレーム１４０２のもとの位置から右下方に移動している。棒線画は現在フレームでは腕を上げ、さらに前フレーム１４０２の中央から右下方にシフトしている。動き予測アルゴリズムを利用することにより、これら２つの映像フレーム１４０２と１４０４との間の変化を適切に表すことができる。

【００９５】

一実施例では、動き予測アルゴリズムは、各輝度マクロブロックに対して総合的な２次元（２Ｄ）空間探索を実行する。実施形態に応じて、動き予測はＭＰＥＧ映像の色光度に直接適用されてはならない。なぜなら、色の動きは輝度と同じ動き情報により適切に表すことができるからである。多くの様々な方法が動き予測の実現に利用可能であるので、動き予測を行う方式はアプリケーションの複雑さと質の問題に依存する。一般に、広い２Ｄ

領域での完全かつ網羅的な探索により、最もよく一致する結果を生成することができる。しかしながら、動き予測はしばしば映像符号化において最も計算量のかかる部分であるので、このパフォーマンスは膨大な計算コストを要する。画素探索の範囲や探索タイプを制限することによるコストを減らす試みは、映像クオリティをある程度犠牲にするというものである。

【００９６】

図１９Ｂは、マクロブロック探索の一例を示す。フレーム１４１０と１４２０は、それぞれ様々なマクロブロックを備えている。現在フレームのターゲットマクロブロック１４３０は、前フレーム１４１０と１４２０からの前マクロブロックと一致した現在マクロブロックである。第１フレーム１４１０では、良好に一致しないマクロブロック１４１２は、案内標識部分を含み、現在マクロブロックと良好に一致していない。第２フレーム１４２０では、符号化対象の現在マクロブロック１４３０と同様に、良好に一致したマクロブロック１４２０は案内標識と傍線画の頭部のビットを含んでいる。これら２つのマクロブロック１４２２と１４３０は、ある程度の共通点を有しており、わずかな違いしか見出すことはできない。相対的に良好な一致が検出されているので、エンコーダは当該マクロブロックに動きベクトルを割り当てる。これらのベクトルは、一致が達成されるように、マクロブロックの水平及び垂直方向への移動量を表す。

【００９７】

図２０は、第２フレームの生成における動き予測の適用例と予測結果を示す。前フレーム１５１０は、時間に関し現在フレーム１５２０の前にくる。本実施例では、符号化及び送信対象のより複雑さを有しない残差エラー画像１５３０を得るために、現在フレーム１５２０が前フレーム１５１０から差し引かれる。本実施例の前フレーム１５１０は、案内標識１５１１と棒線画１５１３から構成される。現在フレーム１５２０は、ボード１５２４上の案内標識１５１１と２つの棒線画１５２２と１５２３から構成される。動きがより正確に予測及び一致すると、残差エラーはより高い確率でゼロに近づき、それにより高い符号化効率が達成される。マクロブロック予測は、探索窓サイズの減少に貢献する。

【００９８】

符号化効率の向上は、動きベクトルがマクロブロック間で大きな相関を有する傾向があるという事実を利用することにより達成することができる。従って、水平要素が、以前に有効な水平方向の動きベクトルと比較され、その差が符号化されてもよい。同じように、符号化前に、垂直要素の差を計算することができる。本実施例では、前フレーム１５１０からの現在フレーム１５２０を差し引くことにより、腕を上げた第２の棒線画１５３２とボード１５３４を含む残差画像１５３０が生成される。この残差画像１５３０は、圧縮そして送信される。この残差画像１５３０は、現在フレーム１５２０全体の圧縮及び送信より、符号化及びより少ないメモリの使用を可能にする複雑さが低減されたものであることが理想的である。しかしながら、必ずしもすべてのマクロブロック探索が許容できる一致を生じさせるとは限らない。エンコーダが許容可能な一致がないと判断すると、特定のマクロブロックが符号化される。

【００９９】

図２１Ａ及び２１Ｂは、動き予測において処理される一例となる現在フレーム１６０１と前フレーム１６５０を示す。前フレーム１６５０は、映像フレーム系列に対し時間順で現在フレーム１６０１を移行していく。各フレームは、フレームにおいて水平及び垂直方向に延びる多数の画素から構成される。現在フレーム１６０１は、水平及び垂直方向に配置されている多数のマクロブロック１６１０、１６２１から１６２７から構成される。本実施例では、現在フレーム１６０１は、同じサイズの重複のないマクロブロック１６１０、１６２１から１６２７に分割される。これら正方形のマクロブロックのそれぞれは、さらに同数の行と列に分割される。同一のマクロブロック１６１０に対し、８行８列のマトリックスを見ることができる。マクロブロック１６１０の各正方形が１つの画素に対応している。従って、このサンプルマクロブロック１６１０は６４画素を含んでいる。他の実施例では、マクロブロックは１６行１６列（１６×１６）のサイズであってもよい。一実

施例では、各画素のデータは、８データビットまたは１ワードから構成されている。他の実施例では、データ画素は、ニブル、ワード、ダブルワード、クアドワードなどを含む他のサイズから構成することができる。現在フレームのこれらの現在マクロブロックは、動き予測のため前フレーム１６５０のマクロブロックと一致させられる。

【０１００】

本実施例では、前フレーム１６５０は、フレームの一部が囲まれた探索窓１６５１を含む。探索窓１６５１は、現在フレーム１６０１からの現在マクロブロックが一致させられる領域を有する。現在フレームと同様に、探索窓は複数のサイズの等しいマクロブロックに分割される。８行８列を有する一例となるマクロブロック１６６０がここでは示されているが、マクロブロックは１６行１６列を有する他の様々なサイズから構成することができる。一実施例の動き予測アルゴリズムにおいて、探索窓１６５１からの各マクロブロックは、許容できる一致を検出するため系列において、現在フレームからの現在マクロブロックと比較される。一実施例では、探索窓１６５１の第１前マクロブロックの左上方のコーナーが探索窓１６５１の左上方コーナーと並べられる。動き予測アルゴリズムにおいて、マクロブロック処理の方向は、画素単位で探索窓の左側から右端に進行する。従って、第２マクロブロックの左端は、探索窓の左端からの１画素である。第１画素行の終わりで、アルゴリズムは探索窓の左端に戻り、次の行の第１画素から進んでいく。探索窓１６５１の各画素のマクロブロックと現在マクロブロックとの比較が完了するまで、この処理は繰り返される。

【０１０１】

図２２Ａから２２Ｄは、本発明の一実施例によるフレームの動き予測の動作を示す。ここで説明される本発明の実施例は、完全な探索動き予測アルゴリズムに関する。完全な探索では、前フレーム（参照フレーム）の探索窓のすべての画素位置に対するマクロブロックが、現在フレームのマクロブロックと一致するよう試みられる。一実施例では、高速完全探索動き予測アルゴリズムは、ＳＩＭＤ右方向シフトマージ処理を利用して、フレームからのＰａｃｋｅｄデータの高速処理を行う。一実施例のＳＩＭＤ右方向シフトマージ処理はまた、データロード数、特に並べられていないメモリロード数及び他のデータ操作命令数を減らすことによって、プロセッサのパフォーマンスを向上させることができる。一般に、一実施例の動き予測処理は以下の擬似コードで記述することができる。

【０１０２】

【表２】

```
x方向及びy方向の各現在ブロックに対し｛
        探索窓のy軸のすべてのmod 1位置に対し｛
                探索窓のx軸のすべてのmod 4位置に対し｛
                        メモリからレジスタに画素データをロード；
                        隣接する４つの前マクロブロックに対しブロック一致処理を実行；
                        当該前マクロブロックの最小値及びインデックス位置を追跡；
}}}
```

ここで、ブロック一致処理では以下が行われる。

【０１０３】

【表３】

```
１からｍの各ラインに対し｛
        第１列から第４列で始まる各マクロブロックに対し｛
                レジスタに保持されるデータから当該前[ライン]の正しいデータを生成；
                評価データ[ライン]+=絶対差（現在[ライン], 前[ライン]）の合計；
}}
```

従って、本実施例では、探索窓の各画素位置に対する前マクロブロックが現在マクロブロックに対し評価される。上述のように、本実施例ではループあたり４つの隣接する前マクロブロックが評価される。画素データはメモリ配置ロードにより、メモリからレジスタにロードされる。右方向シフトマージ処理の利用を通じて、この画素データは操作され、隣接するマクロブロックに適したシフトデータセグメントの様々な組み合わせが生成される。例えば、第１前マクロブロックの第１ラインにおける第１、第２、第３及び第４画素がそれぞれメモリアドレス０、１、２及び３においてスタートすることができる。第２前マクロブロックの第１ラインの第１画素に対して、当該画素はメモリアドレス１において開始される。従って、レジスタデータの右方向シフトマージ処理は、第１前マクロブロックのメモリからすでにロードされたデータを再利用することによって、第２前マクロブロックの必要な画素ラインデータを生成することができ、それによって、時間とリソースの節約が可能になる。同様のシフトマージ処理により、第３、第４、…のような他の隣接する前マクロブロックのラインデータを生成することができる。

【０１０４】

一実施例の動き予測アルゴリズムのブロック一致処理は以下の擬似コードで記述することができる。

【０１０５】

【表４】

```
隣接する４つの前マクロブロックのブロック一致処理｛
１からｍの各ラインに対し｛
        現在マクロブロックの１つのラインの画素データをロード；
        探索窓の１ラインの連続する２つの画素データ群のメモリからレジスタへの並べられた
メモリロード；
        右方向シフトマージ処理によるロードされたデータから隣接する４つの前マクロ
ブロックの各々に対し適切な画素データラインを生成；
        隣接する４つの前マクロブロックの各々に対し前マクロブロックの１ラインと現在マクロ
ブロックの対応するライン間の絶対差の合計の計算；
        隣接する４つの前マクロブロックの各々に対し４つの絶対差の合計を累積；
}}
```

この処理が以下でさらに説明される。これらの実施例は探索窓の隣接する４つのマクロブックにおける処理に関し説明されているが、本発明の他の実施例はそれに制限されるものではない。しかしながら、本発明の実施例は隣接するマクロブロックでの処理に制限されるものではない。処理に要する複数の参照マクロブロックは１画素ずつ変わる必要はない。一実施例では、特定の画素位置の周囲の１６×１６の窓の中の画素を有する任意の参照マクロブロックを一緒に処理してもよい。利用可能なデータレジスタ及び実行ユニットのようなハードウェアリソースの大きさに応じて、他の実施例はブロック一致処理やマクロ

ブロック数に関する絶対差の和の計算を実行することもできる。例えば、８データセグメント幅の２つのデータ群に対する右方向シフトマージ処理から生成される画素データの４つの異なる組み合わせを保持する少なくとも８つのＰａｃｋｅｄデータレジスタを有する他の実施例では、４つの隣接する前マクロブロックへの２つの並べられた８データセグメント幅のメモリロードによる処理を行うことが可能である。８つのＰａｃｋｅｄデータレジスタのうちの４つが、前フレームからの最初の８データセグメント、前フレームの次の８データセグメント、現在フレームのための８データセグメント、及び右方向シフトマージしょりからの８データセグメントを保持する計算オーバヘッドに利用される。その他の４つのＰａｃｋｅｄデータレジスタは、４つのマクロブロックのそれぞれに対し、絶対差の和（ＳＡＤ）の合計を累積するのに利用される。より多くのＰａｃｋｅｄデータレジスタが、共に処理される参照マクロブロックの数を増やすために、ＳＡＤ計算と累積計算のために加えられてもよい。従って、４つの追加的Ｐａｃｋｅｄデータレジスタが利用可能である場合、４つの追加的な前マクロブロックもまた処理可能となる。一実施例の累積された絶対差の和を保持するのに利用可能なＰａｃｋｅｄデータレジスタ数は、一度に処理可能なマクロブロック数を制限するかもしれない。

【０１０６】

さらに、いくつかのプロセッサアーキテクチャでは、メモリアクセスは特定の粒度を有し、ある境界で並べられる。例えば、あるプロセッサは１６または３２バイトブロックに基づきメモリにアクセスすることができる。この場合、１６または３２バイトの境界で揃えられていないデータへのアクセスには、揃えられていないメモリアクセスを必要とし、実行時間とリソースにおいてコストがかかってしまう。さらに悪いことに、所望のデータ部分が境界をまたぎ、複数のメモリブロックを重複させてしまうかもしれない。異なる２つのキャッシュラインにあるデータにアクセスするために、揃えられていないロードを要するキャッシュライン分割はコストがかかってしまう。メモリページの境界をまたぐデータラインはさらに悪い。例えば、８バイトメモリブロック及び画素あたり１バイトのデータを有する８画素にわたるマクロブロックにより動作するプロセッサでは、１つの揃えられたメモリロードで当該マクロブロックラインにとって十分であろう。次の隣接マクロブロックがなければ、その画素ラインに必要なデータの１画素列は、第１マクロブロックからの、しかしまた、次のメモリブロックの１データバイトのメモリ境界を越してメモリブロックの７データバイトにわたるであろう。本発明の実施例は、右方向シフトマージ処理を使って、効率的にデータを処理する。一実施例では、２つの連続したメモリブロックが揃えられたメモリ境界に置かれ、いくつかの利用のためにレジスタに保持される。右方向シフトマージ処理は、これらのメモリブロックを取得し、そのなかのデータセグメントを正しいデータラインを得るのに必要な距離だけシフトする。本実施例によると、右方向シフトマージしょりは、ロード済みの２つのメモリブロックを取得し、第２ブロックから１データバイトをシフトし、第１ブロックから第２ブロックへ１データバイトをシフトし、並べられていないロードを実行する必要なく第２マクロブロックの第１ラインのデータを生成する。動き予測の実施例はまた、アルゴリズムの実現方法に基づき従属関係の連鎖を断ち切ることもできる。例えば、計算順序を変えることによって、データ／命令の依存関係が、図１５のプロセッサ１０００と同じように、ある計算及び命令を順序に従わずに削除あるいはシフトすることが可能である。実行待ち時間と利用可能な計算リソースの増加によって、より新しい世代のプロセッサアーキテクチャによるパフォーマンスの向上がより大きくなる。右方向シフトマージ処理の一実施例を使うことによって、ブロック一致シーケンスにおけるある依存関係が回避することができる。例えば、複数の絶対差の和の演算及び／あるいは累積演算は並列に実行することができる。

【０１０７】

図２２Ａは、現在フレーム１７０１における現在マクロブロックの経過を示す。本実施例では、各現在マクロブロック１７１０は１６行１６列に分割され、それによって、２５６の画素から構成されるようになる。本実施例では、各マクロブロック１７１０の画素は一度に各列１７１１処理される。現在ブロックの１６行すべてが探索窓の所望のマクロブ

ロックに対し処理されると、次の現在マクロブロックが処理される。本実施例のマクロブロックは、マクロブロックサイズ毎に現在フレーム１７０１の左側から右側に水平方向１７２０で処理される。言い換えると、本実施例において現在マクロブロックは重複することなく、現在マクロブロックは各マクロブロックが次のマクロブロックと隣接するよう配置される。例えば、第１マクロブロックは第１画素列から第１６画素列に拡張可能である。第２マクロブロックは第１７列から第３２列に、以下同様にして、拡張することができる。マクロブロック行の終わりで、処理は左端に戻り（１７２２）、本実施例では１６行である１マクロブロックの高さだけ下に降りる。フレーム１７０１１の全体に対し一致処理が完了されるまで、マクロブロックサイズ毎にマクロブロックは水平方向に左から右へ処理されていく（１７２４）。

【０１０８】

図２２Ｂは、前（参照）フレームの探索窓１７５１におけるマクロブロックの経過を示す。特定の実施形態に応じて、探索窓１７５１はある領域に焦点が当てられ、前フレーム全体よりも小さくされる。他の実施例では、探索窓は前フレームに完全に重複してもよい。現在フレームと同じように、各前マクロブロック１７６０、１７６５、１７７０及び１７７５は、それぞれが合計で２５６画素の１６行１６列に分割される。本実施例では、探索窓１７５１の４つの前マクロブロック１７６０、１７６５、１７７０及び１７７５が、一致探索での１つの現在ブロックに対して並列に処理される。現在フレームの現在マクロブロックと異なり、探索窓１７５１の前マクロブロック１７６０、１７６５、１７７０及び１７７５は本実施例と同じように重複しうる。ここで、各前マクロブロックは１画素列だけシフトされる。従って、ＢＬＫ１の第１行の最左画素は画素１７６１であり、ＢＬＫ２では画素１７６６、ＢＬＫ３では画素１７７１、ＢＬＫ４では画素１７７６である。動き予測アルゴリズムでは、前マクロブロック１７６０、１７６５、１７７０及び１７７５の各行は現在ブロックの対応する行に対して比較される。例えば、ＢＬＫ１１７６０、ＢＬＫ２１７６５、ＢＬＫ３１７７０及びＢＬＫ４１７７５の第１行はそれぞれ現在ブロックの第１行と共に処理される。

【０１０９】

マクロブロックの１６行すべてに対し実行されるまで、４つの重複した隣接するマクロブロックに対する行単位での比較が続けられる。本実施例のアルゴリズムは、次の４つのマクロブロックでの処理のため、４画素列だけシフトする。従って、例えば、次の４つのマクロブロックの最左第１画素列は、それぞれ画素１７９６、１７９７、１７９８及び１７９９となる。本実施例では、探索窓が完了するまで、前マクロブロック処理は探索窓１７５１において右方向１７８０へ続けられ、探索窓１７５１の最左画素において１画素列下に降りて再開される。本実施例の現在フレームの現在マクロブロックは重複せず、次のマクロブロックはマクロブロックの高さあるいは幅であるが、前フレームまたは参照フレームの前マクロブロックは重複し、次のマクロブロックは１画素行あるいは１画素列だけインクリメントされる。本実施例の４つの基準マクロブロック１７６０、１７６５、１７７０及び１７７５は隣接し、１画素列だけ異なっているが、選択された画素位置の周囲の特定領域を重複している探索窓１７５１の任意のマクロブロックを当該画素位置のマクロブロックとともに処理することができる。例えば、画素１７９６のマクロブロック１７６０が処理される。画素１７９６の周囲の１６×１６の窓の中の任意のマクロブロックを、マクロブロック１７６０と共に扱うことができる。本実施例の１６×１６の窓は、マクロブロックのサイズと行のライン長による。この場合、１つの行またはデータラインは、１６のデータ要素を有する。動き予測アルゴリズムの本実施例のこのブロック一致機能により、１６のデータ要素の２つのデータラインがロードされ、右方向シフトマージ処理の実行により２つのデータラインのシフト／マージされたものを有する様々なデータラインが生成されるので、このマクロブロックに対しデータがロードされる１６×１６の窓を重複する他のマクロブロックがこのロードされたデータを少なくとも部分的に再利用することができる。従って、マクロブロック１７６５、１７７０及び１７７５のような、マクロブロック１７６０を重複する任意のマクロブロック、あるいはマクロブロック１７６０の右

下端画素位置で始まるマクロブロックを、マクロブロック１７６０と共に処理することができる。重複の大きさの差は、前のデータロードから再利用することができるデータ量に影響を与える。

【０１１０】

本発明による動き予測の実施例によると、マクロブロック解析は、２つのマクロブロック間の絶対差の和を得るために、行単位により前（参照）マクロブロックと現在マクロブロックとの比較からなる。この絶対差の和は、これらのマクロブロックがどれくらい異なっているか、そしてどれくらい近い一致が存在するかについて示している。一実施例の各前マクロブロックは、当該マクロブロックの１６行すべての絶対差の和を累積することにより得られる値により表すことができる。解析中の現在マクロブロックに対し、最も近く一致するマクロブロックの記号が維持される。例えば、絶対差の最小累積和と、対応する前マクロブロックの位置インデックスが追跡される。動き予測が探索窓において進むとき、各前マクロブロックの累積和とこの最小値が比較される。より新しい前マクロブロックが追跡されている最小値のものより小さい累積差を有し、すなわち、これまでに最も近い一致よりもさらに近い一致を示すものがある場合、この新しい前マクロブロックの累積差とインデックス上方が、新しい最小差とインデックスになる。探索窓のずべての画素に対する利用可能なマクロブロックが一実施例において処理されるとき、最小差を有するこのインデックスのマクロブロックが、当該現在フレームの圧縮のため残差画像を取得するのに利用される。

【０１１１】

図２２Ｃは、本発明の一実施例による現在ブロック１８４０による所与の探索窓の４つの参照マクロブロック１８１０、１８１５、１８２０及び１８２５の並列処理を示す。本実施例では、探索窓の画素データが「Ａ、Ｂ、Ｃ、Ｄ、Ｅ、Ｆ、Ｇ、Ｈ、Ｉ、Ｊ、Ｋ、Ｌ、Ｍ、Ｎ、Ｏ、Ｐ」１８６０として順序付けされている。ここで、「Ａ」はデータセットにおける最下位アドレス位置（０）であり、「Ｐ」は最上位アドレス一（１５）である。この画素セット１８６０は、各自が８（ｍ）データセグメントを有する２つのせく著ン１６８１と１６８２から構成される。右方向シフトマージ処理により、上述のように、本発明の実施例はこれら２つのデータセクション１６１８と１６８２によりオペランドを操作し、異なる前マクロブロック１８１０、１８１５、１８２０及び１８２５に対し適切に並べられた行データ１８３０を生成することができる。前マクロブロック１８１０、１８１５、１８２０及び１８２５と現在マクロブロック１８４０の各マクロブロックは、ｍ行ｍ列のサイズを有する。説明及び簡単化のため、本実施例においてｍは８に等しい。他の実施例は、異なるサイズのマクロブロックを有することができ、例えば、ｍは４、１６、３２、６４、１２８、２５６などであってもよい。

【０１１２】

本実施例では、動き予測アルゴリズムが、これら４つの前ブロック１８１０、１８１５、１８２０及び１８２５の第１行に、現在ブロックの第１行と共に適用される。一実施例では、２つのマクロブロック幅（２ｍ）に対し２つのデータセクション１８６１と１８６２を含む画素データが、２つの配置メモリロード命令によりメモリからロードされ、仮レジスタに保持される。この２つのデータセクション１８６１と１８６２への右方向シフトマージ処理により、メモリアクセスを多用することなく、行データ１８３０の可能な９つの組み合わせを生成することができる。さらに、実行時間とリソースを多用する非配置メモリロードを回避することが可能になる。本実施例では、２つのデータセクション１８６１と１８６２がバイト境界で並べられる。例えば、データセグメントＢまたはＤのようなバイト境界でのアドレスから始まらないメモリロードは、典型的には、非配置メモリロード処理を要するであろう。各ブロックの行データ１８３０は以下のようになる。ここで最左データセグメントが最下位アドレスである。ＢＬＯＣＫ１１８１０において、ＲＯＷ１１８１１は、「Ａ、Ｂ、Ｃ、Ｄ、Ｅ、Ｆ、Ｇ、Ｈ」から構成される。ＲＯＷ１１８１１のデータが第１データセクション１８６１と同じであるとき、シフト処理は必要ない。しかし、ＢＬＯＣＫ２１８１５のＲＯＷ１１８１６は、「Ｂ、Ｃ、Ｄ、Ｅ、Ｆ、Ｇ、Ｈ、Ｉ」

から構成されている。前ＢＬＯＣＫ１
１８１０とＢＬＯＣＫ２１８１５が水平方向に１画素だけずれているので、ＢＬＯＣＫ２１８１５は画素データＢから始まるが、ＢＬＯＣＫ１１８１０は画素データＡから始まり、第２画素データはＢである。従って、１のシフトカウントにより、２つのデータセクション１８６１と１８６２の右方向シフトマージ処理により、ＢＬＯＣＫ２ＲＯＷ１データが生成される。

【０１１３】

同様にして、ＢＬＯＣＫ３１８２０は、右方向へさらに１画素シフトしており、ＢＬＯＣＫ３１８２０のＲＯＷ１１８２１は、画素データＣから始まり、「Ｃ、Ｄ、Ｅ、Ｆ、Ｇ、Ｈ、Ｉ、Ｊ」から構成される。２のシフトカウントによる２つのデータセクション１８６１と１８６２のオペランドに対する右方向シフトマージ処理により、ＢＬＯＣＫ３ＲＯＷ１データが生成される。ＢＬＯＣＫ４１８２５のＲＯＷ１１８２６は、「Ｄ、Ｅ、Ｆ、Ｇ、Ｈ、Ｉ、Ｊ、Ｋ」から構成される。このデータは、同じデータオペランドに対する４のシフトカウントによる右方向シフトマージ処理により生成される。一時的にセーブされ、前にロードされたデータセクション１８６１と１８６２に対する右方向シフトマージ処理により、他の隣接するマクロブロックの行データの生成におけるデータの再利用と、メモリロード数、特に非配置メモリロード数を減らすことによる時間／リソースの節約が可能になる。ここで、現在ブロックの画素データは、前フレームの参照マクロブロックに対する絶対差の和のすべての比較において同じである。１つの配置メモリロードは、現在ブロック１８４０がメモリ境界で並んでいるとき、現在ブロック１８４０の行データ１８４２に対し可能であるかもしれない。

【０１１４】

動き予測の一実施例のこの例に関して、絶対差の和を得るために、前マクロブロック１８１０、１８１５、１８２０及び１８２５の各行が現在ブロック１８４０の対応する行と比較される。従って、ＢＬＯＣＫ１１８１０のＲＯＷ１１８１１が、絶対差の和（ＳＡＤ）を求める処理１８５０において、現在ブロック１８４０のＲＯＷ１１８４１と比較される。ここで処理されている他の３つのブロックに関しても同様の処理が行われる。これら４つのマクロブロック１８１０、１８１５、１８２０及び１８２５は同時あるいは並列に処理されているようであるが、本発明の他の実施例はこれに限定されない。従って、この４つのマクロブロックの処理は逐次的に４つのシーケンスとして実行することができる。例えば、各参照ブロックの第１行は、ＢＬＯＣＫ１１８１０、ＢＬＯＣＫ２１８１５、ＢＬＯＣＫ３１８２０及びＢＬＯＣＫ４１８２５の順で、現在ブロック１８４０と共にＳＡＤ処理１８５０が実行される。その後、各参照ブロックの第２行に対し、そして以下同様に、ＳＡＤ処理１８５０が実行される。各ＳＡＤ処理１８５０の実行後、絶対差の和の現在トータルが仮レジスタに蓄積される。従って、本実施例では、４つのレジスタが、当該マクロブロックのすべてのｍ行が処理されるまで、絶対差の和を蓄積していく。マクロブロックの最良一致探索の一部として、各ブロックに対して蓄積された値がこれまでの最小差と比較される。本実施例は４つの隣接しかつ重複した前マクロブロックの処理を説明しているが、探索窓の第１ブロックＢＬＫ１８１０に重複している他のマクロブロックもまた、データラインに関連がある場合、ＢＬＫ１８１０のデータラインと共に処理することができる。従って、処理中のマクロブロックの周囲の１６×１６の窓の中のマクロブロックもまた処理することができる。

【０１１５】

図２２Ｄは、絶対差の和（ＳＡＤ）の計算処理１９４０とこれらＳＡＤ値の合計処理を示す。ここで、参照マクロブロックＢＬＯＣＫ１１９００のＲＯＷ　ＡからＲＯＷ　Ｐの各行と、現在マクロブロック１９２０におけるそれに対応するものとが、ＳＡＤ処理１９４０を実行される。ＳＡＤ処理１９４０では、各行の画素を表すデータが比較され、前マクロブロック１９００と現在マクロブロック１９２０からの２つの行の間の絶対差を表す値が計算される。ＡからＰのすべての行に対して、ＳＡＤ処理１９４０による値がブロック和１９４２として合計される。このブロック和１９４２は、前マクロブロック１９００

と現在マクロブロック１９２０全体に対する絶対差の和の蓄積された値を提供する。このブロック和１９４２に基づき、動き予測アルゴリズムは、前マクロブロック１９００が現在マクロブロック１９２０に関してどのくらい近い一致であるか決定することができる。

【０１１６】

本実施例は一度に４つの参照マクロブロックに対して動作するが、他の実施例はロードされる画素データ量及び利用可能なレジスタ数に応じて、異なる数のマクロブロックに対して動作することができる。さらに、動き予測処理では、様々なレジスタを利用することができる。例えば、ＭＭＸ技術によるｍｍレジスタやＳＳＥ２技術によるＸＭＭレジスタのような拡張レジスタを利用して、画素データのようなＰａｃｋｅｄデータを保持することができる。一実施例では、６４ビット幅のＭＭＸレジスタは８バイトを保持することができ、またもし各画素が８ビットデータを有していれば、８画素を保持することができる。他の実施例では、１２８ビット幅のＸＭＭレジスタは、１６バイト、もし各画素が８ビットデータであれば、１６画素を保持することができる。同じように、Ｐａｃｋｅｄデータを保持する３２／１２８／２５６／５１２ビット幅のような他のサイズのレジスタもまた、本発明の実施例と共に利用可能である。他方、通常の整数演算のようなＰａｃｋｅｄデータレジスタを必要としない計算は、整数レジスタや整数ハードウェアを利用することができる。

【０１１７】

図２３Ａは、動きを予測する方法の一実施例を示すフローチャートである。ブロック２００２において、追跡される最小値（ｍｉｎ）と、この最小値に対するインデックス位置が初期化される。本実施例では、この追跡されているｍｉｎ値とインデックスは、探索窓からの処理された前（参照）マクロブロックのどれが現在マクロブロックに最も近く一致しているか示す。ブロック２００４において、現在フレームのすべての所望のマクロブロックが完了されたかチェックされる。もし完了されていれば、動き予測アルゴリズムのこの部分が実行される。もしすべての所望の現在マクロブロックが処理されていなければ、ブロック２００６において未処理の現在マクロブロックが現在フレームに対し選択される。ブロック２００８において、ブロック一致処理が前（参照）フレームの探索窓における第１画素位置から進む。ブロック２０１０において、探索窓が完了されたかチェックされる。第１のパスでは、探索窓の何れもが処理されていない。しかし次のパスで、探索窓の全体が処理されていれば、フローはブロック２００４に戻り、他の現在マクロブロックが利用可能であるか判断される。

【０１１８】

探索窓の全体が解析されていなければ、ブロック２０１２において、このＸ軸行に沿ったすべての画素が処理されているかチェックされる。当該行が処理されていれば、行カウントは次の行にインクリメントされ、フローはブロック２０１０に戻り、この新たな行においてより多くのマクロブロックが探索窓において利用可能であるかチェックされる。しかしこの行の画素のすべての利用可能なマクロブロックが処理されていなければ、ブロック２０１４において、この画素列と行におけるマクロブロックが処理されているかチェックされる。もし当該マクロブロックが処理されていれば、列カウントがインクリメントされ、フローはブロック２０１２に戻り、この新しい列における画素のマクロブロックが処理されているかどうかチェックされる。一方、もしこの列及び行の画素のマクロブロックが処理されていなければ、ブロック一致処理がこの参照マクロブロックと現在マクロブロックとの間で実行される。

【０１１９】

本実施例におけるフローは、簡単化のため一度に１画素だけＸとＹ軸に沿って画素の行及び列位置をインクリメントする処理を述べている。しかしながら、本発明の一実施例では、パスあたり４つの前マクロブロックが処理される。従って、Ｙ軸に沿った列カウントはパスあたり４列分インクリメントされる。他の実施例はまた、一度に８、１６、３２などのマクロブロックを処理してもよく、これにより列カウントは、アルゴリズムの後続のパスに対し正しい画素位置を指すよう８、１６、３２などの列だけ対応してインクリメン

トされる。本実施例のブロック一致処理は順序付けされた形式でＸ及びＹ軸に沿って探索が実行されるが、他の実施例のブロック一致処理は、異なるパターンを使うダイアモンド探索やログ探索のような他のアルゴリズムを利用することもできる。

【０１２０】

図２３Ｂは、図２３Ａのブロック一致処理をさらに説明するフローチャートである。ブロック２２２２において、参照マクロブロックと現在マクロブロックのデータがロードされる。一実施例では、２つのＰａｃｋｅｄデータ群が多数の連続する画素のデータを含むとき、参照マクロブロックデータがロードされる。一実施例では、各Ｐａｃｋｅｄデータ群は８つのデータセグメントから構成される。ブロック２２２４において、正確なデータ群を取得するために、右方向シフトマージ処理はデータ群に対し必要なものとして実行される。４つの前マクロブロックが一緒に処理された前述の実施例では、右方向シフトマージ処理は各マクロブロックにあるラインに対応するデータ群を生成することができる。各隣接マクロブロックに対するデータ群が１画素だけシフトされ、マクロブロックは探索窓の各画素行に対し同時に１画素だけスライドしているように見える。ブロック２２２６、２２２８、２２３０及び２２３２における処理が一緒に処理される４つの前マクロブロックのそれぞれに対し適用される。一実施例において、次の処理が実行される前に、４つのすべてのマクロブロックに対し同様の処理が行われる。他の実施例では、１つの前マクロブロックは、適切にシフトされたデータセグメントを含むデータ群を有する次の前マクロブロックが処理される前に、すべての処理を完了させるようにしてもよい。

【０１２１】

ブロック２２２６において、前マクロブロックと現在マクロブロックの対応するライン間の絶対差の和が、これらマクロブロックの各行に対し計算される。ブロック２２２８において、前マクロブロックのすべてのラインに対する絶対差の和が蓄積される。ブロック２２３０において、当該前マクロブロックに対し蓄積された差は現在の最小値と比較される。ブロック２２３２においてこの前マクロブロックの差が現在最小値より小さい場合、最小値はこの新しい差により更新される。インデックスもまた、このマクロブロックがこれまでで最も近い一致であるということを示すために、当該マクロブロックの位置を反映するよう更新される。しかし、ブロック２２３２においてこの新しい差が現在の最小値より大きい場合、当が前マクロブロックはこれまで一致してきたものよりより近い一致ではない。

【０１２２】

本発明による動き予測アルゴリズムの実施例はまた、現在のハードウェアリソースによるプロセッサ及びシステムのパフォーマンスの向上を可能にする。しかしながら、技術の進歩に従って、より多くのハードウェアリソースやより高速でより効率的な論理回路と組み合わされるとき、本発明の実施例はパフォーマンスの向上にさらなる著しい影響を与えうる。従って、動き予測の効率的な一実施例はプロセッサの生成に異なるより大きな影響をもたらしうる。最新のプロセッサアーキテクチャにより多くのリソースを単に付加するだけでは、よりよいパフォーマンスの向上は保証されない。動き予測と右方向シフトマージ命令（ＰＳＲＭＲＧ）の一実施例と同様にアプリケーションの効率を維持することにより、より大きなパフォーマンスの向上が可能になる。

【０１２３】

議論の簡単化のため、上記実施例は一般的に６４ビット幅ハードウェア／レジスタ／オペランドに関し説明されているが、他の実施例では１２８ビット幅ハードウェア／レジスタ／オペランドを利用して、レジスタマージ処理、右方向シフトマージ処理、及び動き予測計算が実行される。さらに、本発明の実施例はＭＭＸ／ＳＳＥ／ＳＳＥ２技術のような特定のハードウェアあるいは技術タイプに制限されるものではなく、他のＳＩＭＤ実施形態及び他のグラフィカルなデータ操作技術と共に利用することもできる。図２０から２３Ｂに関し説明された動き予測及びブロック一致の実施例は、８画素幅または８データ要素幅のライン／行、及び８行８列のマクロブロックに関して説明されているが、他の実施例は他のサイズを含んでいる。例えば、ライン／行は１６画素幅または１６データ要素幅で

ありうるし、マクロブロックは１６行１６列でありうる。

【０１２４】

上記明細書では、本発明は特定の実施例を参照することにより説明されてきた。しかしながら、添付されたクレームに与えられるように、発明のより広範な趣旨及び範囲から逸脱することなく様々な修正及び変更が可能であるということは明らかであろう。従って、明細書及び図面は限定的な意味よりも例示的な意味でみなされるべきである。

【図面の簡単な説明】

【０１２５】

【図１】図１は、本発明の一実施例を実現することができるコンピュータシステムを示すブロック図を示す。

【図２】図２は、本発明のさらなる一実施例による図１に示されるようなプロセッサの一実施例を示すブロック図を示す。

【図３】図３は、本発明のさらなる一実施例によるＰａｃｋｅｄデータタイプを示すブロック図を示す。

【図４Ａ】図４Ａは、本発明の一実施例によるイン・レジスタＰａｃｋｅｄバイト表現を示す。

【図４Ｂ】図４Ｂは、本発明の一実施例によるイン・レジスタＰａｃｋｅｄワード表現を示す。

【図４Ｃ】図４Ｃは、本発明の一実施例によるイン・レジスタＰａｃｋｅｄダブルワード表現を示す。

【図５】図５は、本発明の一実施例によるバイトシャッフル命令の動作を示すブロック図を示す。

【図６】図６は、本発明の一実施例によるバイト乗加算命令を示すブロック図を示す。

【図７Ａ】図７Ａは、本発明のさらなる一実施例による複数の積和ペアを生成する図６に示されるようなバイト乗加算命令と合成された図５のバイトシャッフル命令を示すブロック図を示す。

【図７Ｂ】図７Ｂは、本発明のさらなる一実施例による複数の積和ペアを生成する図６に示されるようなバイト乗加算命令と合成された図５のバイトシャッフル命令を示すブロック図を示す。

【図７Ｃ】図７Ｃは、本発明のさらなる一実施例による複数の積和ペアを生成する図６に示されるようなバイト乗加算命令と合成された図５のバイトシャッフル命令を示すブロック図を示す。

【図８Ａ】図８Ａは、本発明のさらなる一実施例による隣接加算命令を示すブロック図を示す。

【図８Ｂ】図８Ｂは、本発明のさらなる一実施例による隣接加算命令を示すブロック図を示す。

【図８Ｃ】図８Ｃは、本発明のさらなる一実施例による隣接加算命令を示すブロック図を示す。

【図８Ｄ】図８Ｄは、本発明のさらなる一実施例による隣接加算命令を示すブロック図を示す。

【図９Ａ】図９Ａは、本発明のさらなる一実施例によるレジスタマージ命令を示す。

【図９Ｂ】図９Ｂは、本発明のさらなる一実施例によるレジスタマージ命令を示す。

【図１０】図１０は、本発明の一実施例によるコンテンツデータの効率的なデータ処理のためのフローチャートを示すブロック図を示す。

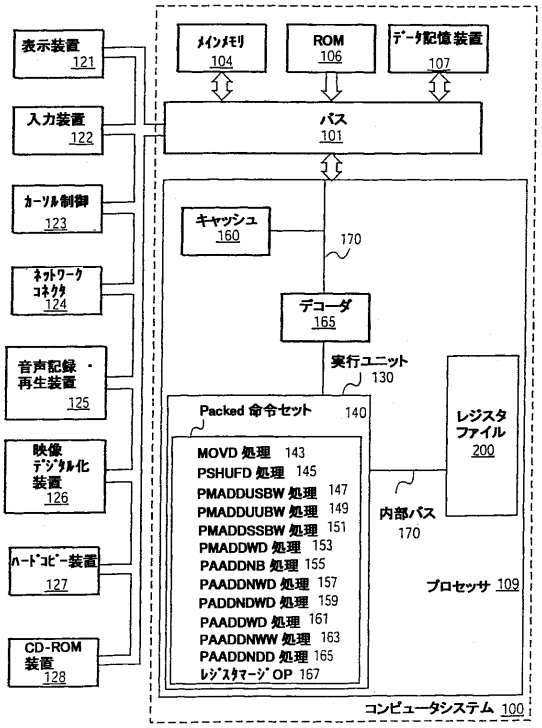
【図１１】図１１は、本発明のさらなる一実施例によるデータ処理によるコンテンツデータを処理するための追加的方法を示すブロック図を示す。

【図１２】図１２は、本発明のさらなる一実施例によるコンテンツデータの継続処理のためのフローチャートを示すブロック図を示す。

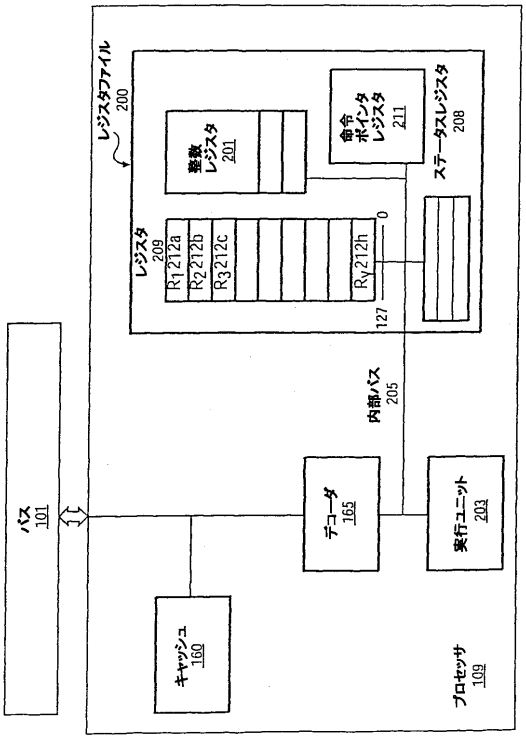
【図１３】図１３は、本発明のさらなる一実施例によるレジスタマージ処理を示すフローチャートを示すブロック図を示す。

【図１４】図１４は、本発明の一実施例によるソースデータ記憶装置からの未処理データ要素を選択するための追加的方法を示すフローチャートを示す。
【図１５】図１５は、本発明による右方向シフトマージ処理を実行する論理回路を含む一実施例のプロセッサのためのマイクロアーキテクチャのブロック図である。
【図１６Ａ】図１６Ａは、本発明によるデータオペランドに対する右方向平行シフトマージ処理を実行する論理の一実施例のブロック図である。
【図１６Ｂ】図１６Ｂは、右方向シフトマージ処理を実行する論理の他の実施例のブロック図である。
【図１７Ａ】図１７Ａは、本発明の第１実施例による右方向平行シフトマージ命令の動作を示す。
【図１７Ｂ】図１７Ｂは、第２実施例による右方向シフトマージ命令の動作を示す。
【図１８Ａ】図１８Ａは、データオペランドを並列に右方向にシフトしマージする方法の一実施例を示すフローチャートである。
【図１８Ｂ】図１８Ｂは、データを右方向にシフトしマージする方法の他の実施例を示すフローチャートである。
【図１９Ａ】図１９Ａは、動き予測の一例を示す。
【図１９Ｂ】図１９Ｂは、動き予測の一例を示す。
【図２０】図２０は、動き予測の適用例と予測結果を示す。
【図２１Ａ】図２１Ａは、動き予測において処理される一例となる現在及び前フレームを示す。
【図２１Ｂ】図２１Ｂは、動き予測において処理される一例となる現在及び前フレームを示す。
【図２２Ａ】図２２Ａは、本発明の一実施例によるフレームに対する動き予測の動作を示す。
【図２２Ｂ】図２２Ｂは、本発明の一実施例によるフレームに対する動き予測の動作を示す。
【図２２Ｃ】図２２Ｃは、本発明の一実施例によるフレームに対する動き予測の動作を示す。
【図２２Ｄ】図２２Ｄは、本発明の一実施例によるフレームに対する動き予測の動作を示す。
【図２３Ａ】図２３Ａは、動き予測方法の一実施例を示すフローチャートである。
【図２３Ｂ】図２３Ｂは、動き予測方法の一実施例を示すフローチャートである。

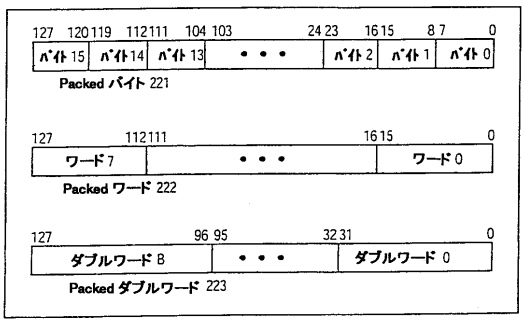
【図１】


表示装置 121
入力装置 122
ｶｰｿﾙ制御 123
ﾈｯﾄﾜｰｸ ｺﾈｸﾀ 124
音声記録・再生装置 125
映像 ﾃﾞｼﾞﾀﾙ化 装置 126
ﾊｰﾄﾞｺﾋﾟｰ装置 127
CD-ROM 装置 128
メインメモリ 104
ROM 106
ﾃﾞｰﾀ記憶装置 107
バス 101
キャッシュ 160
170
デコーダ 165
実行ユニット 130
Packed 命令セット 140
MOVD 処理 143
PSHUFD 処理 145
PMADDUSBW 処理 147
PMADDUUBW 処理 149
PMADDSSBW 処理 151
PMADDWD 処理 153
PAADDNB 処理 155
PAADDNWD 処理 157
PADDNDWD 処理 159
PAADDWD 処理 161
PAADDNWW 処理 163
PAADDNDD 処理 165
ﾚｼﾞｽﾀﾏｰｼﾞ OP 167
レジスタ ファイル 200
内部バス 170
プロセッサ 109
コンピュータシステム 100

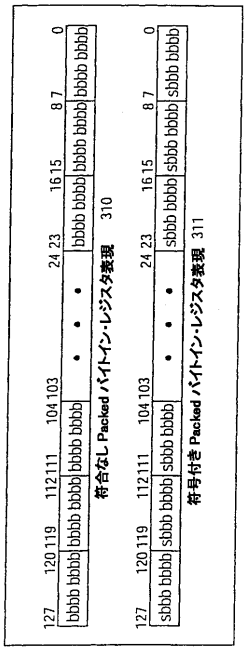
【図２】


レジスタファイル 200
整数 レジスタ 201
命令 ポインタ レジスタ 211
ステータスレジスタ 208
レジスタ 209
R1 212a
R2 212b
R3 212c
Ry 212h
127
内部バス 205
バス 101
デコーダ 165
実行ユニット 203
キャッシュ 160
プロセッサ 109

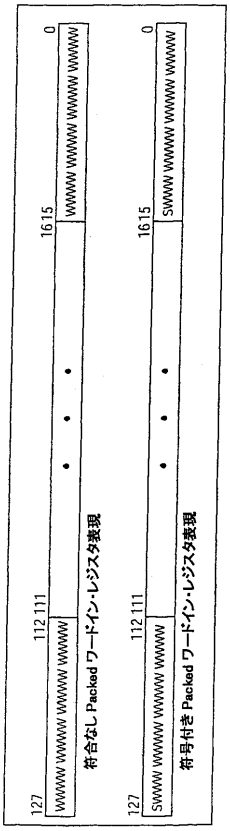
【図３】


Packed バイト 221
Packed ワード 222
Packed ダブルワード 223

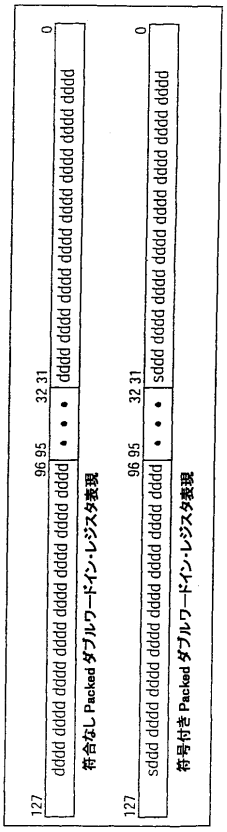
【図４Ａ】


符合なし Packed バイトイン・レジスタ表現 310
符号付き Packed バイトイン・レジスタ表現 311

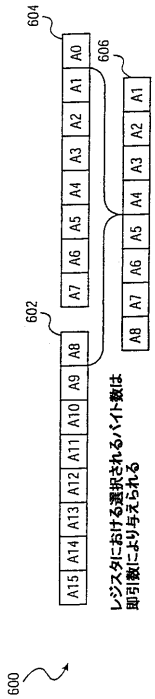
【図４Ｂ】



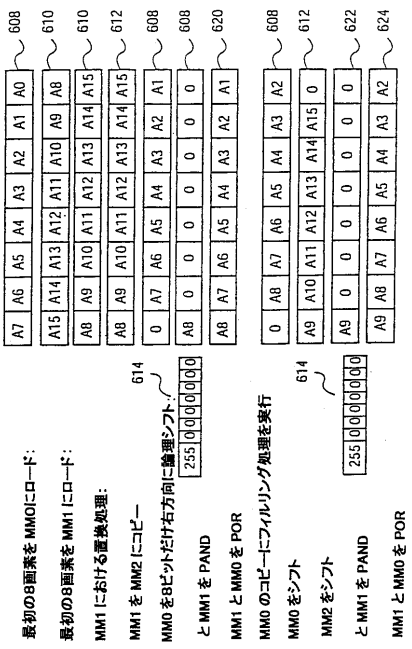
【図４Ｃ】



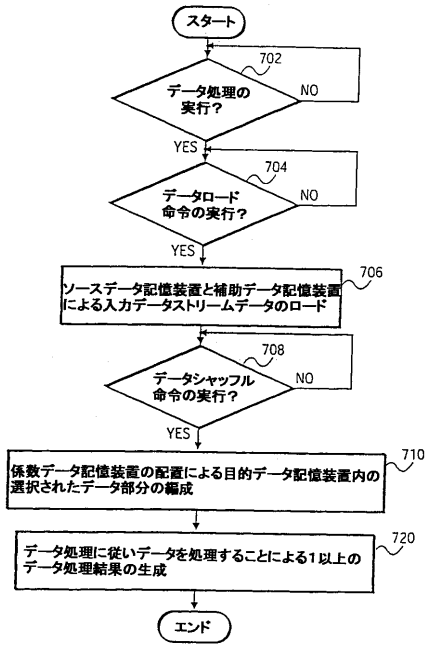
【図９Ａ】



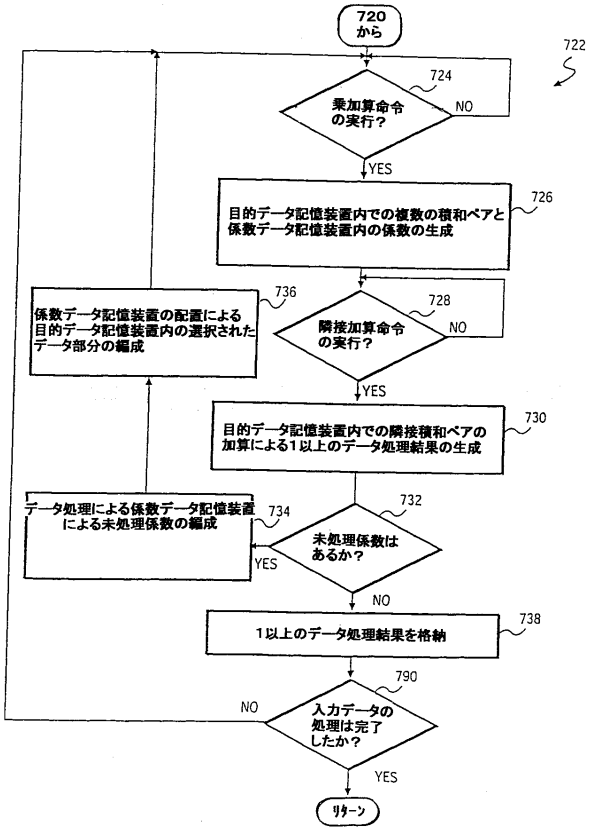
【図９Ｂ】



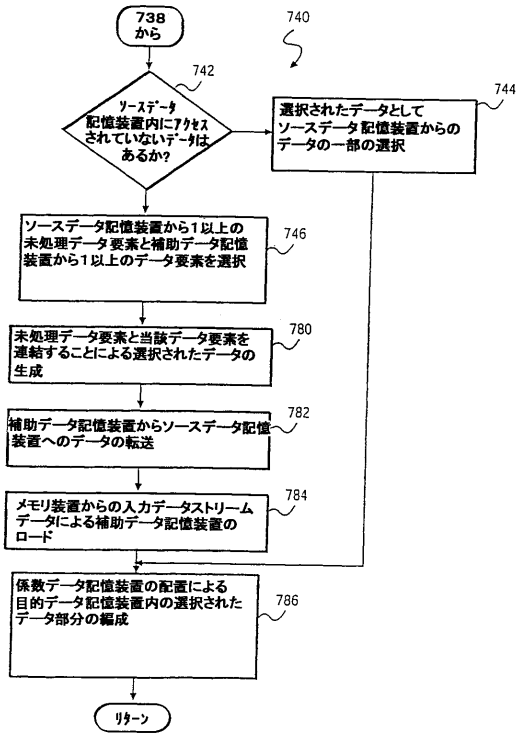
【図１０】



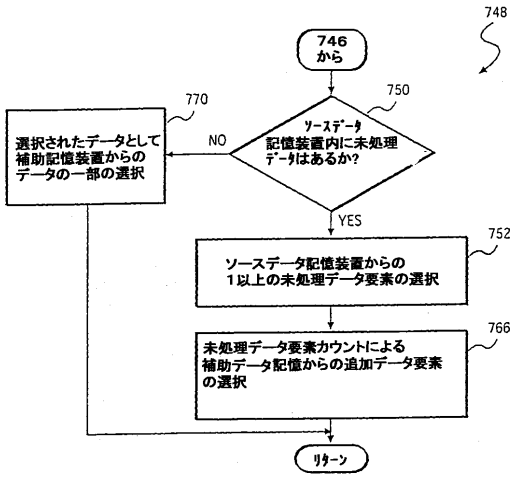
【図１１】



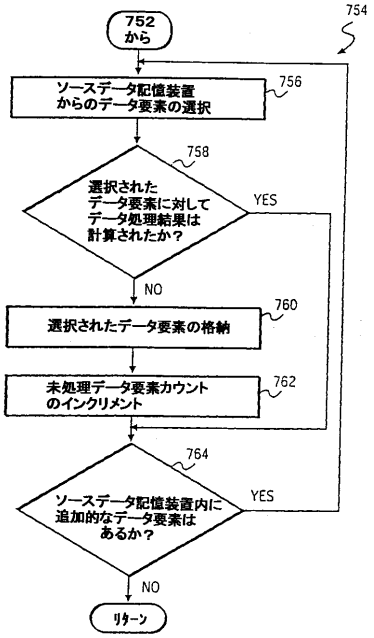
【図１２】



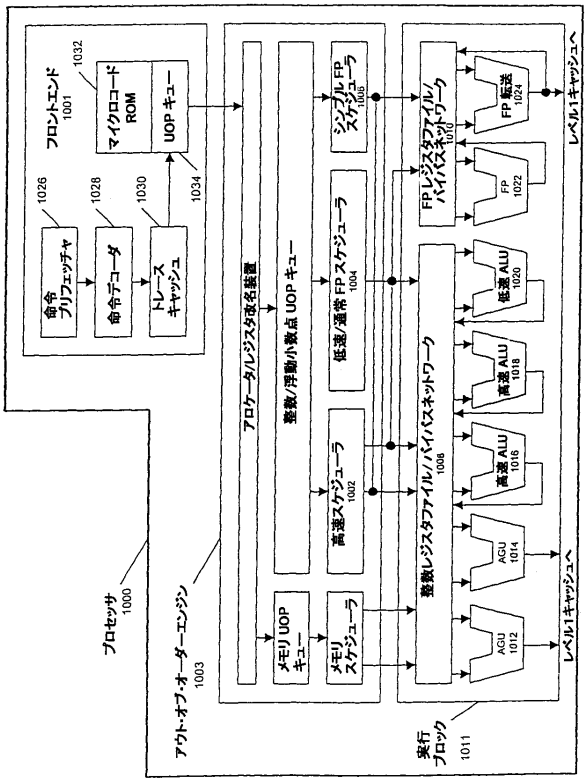
【図１３】



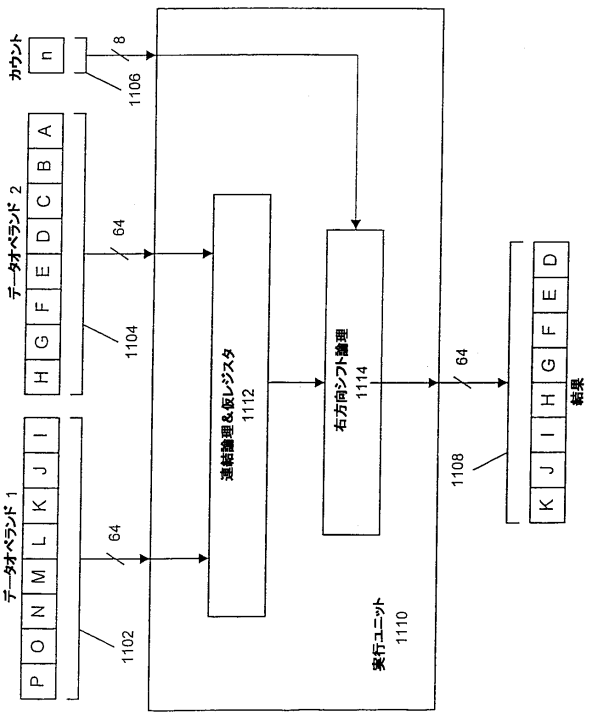
【図１４】



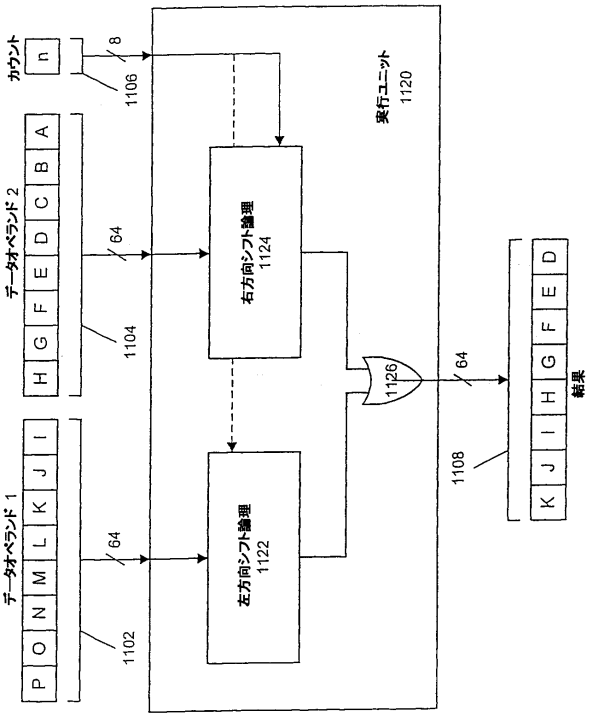
【図１５】



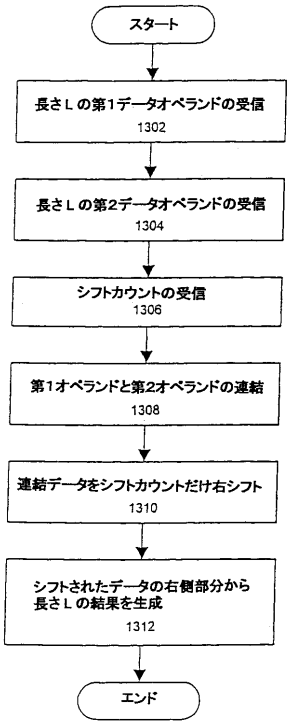
【図１６Ａ】



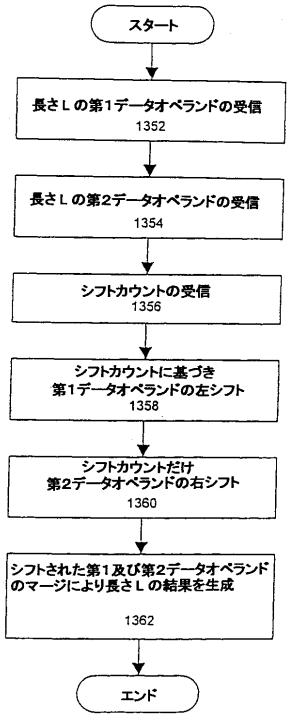
【図１６Ｂ】



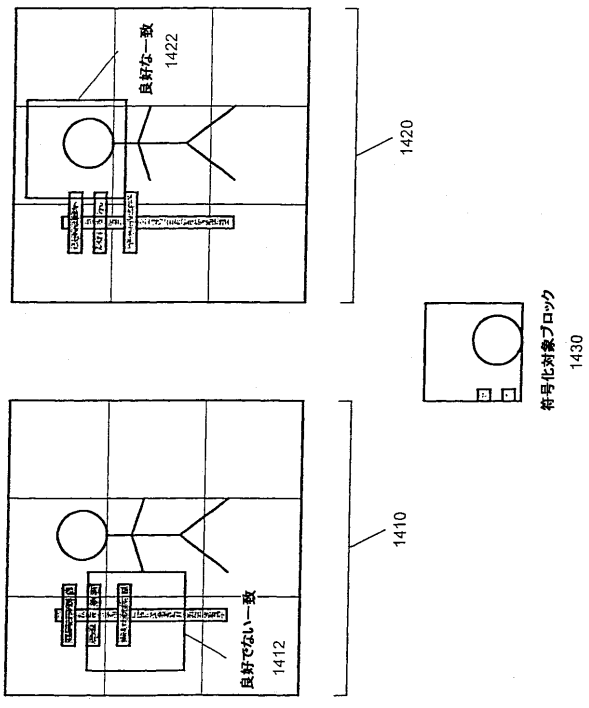
【図１８Ａ】



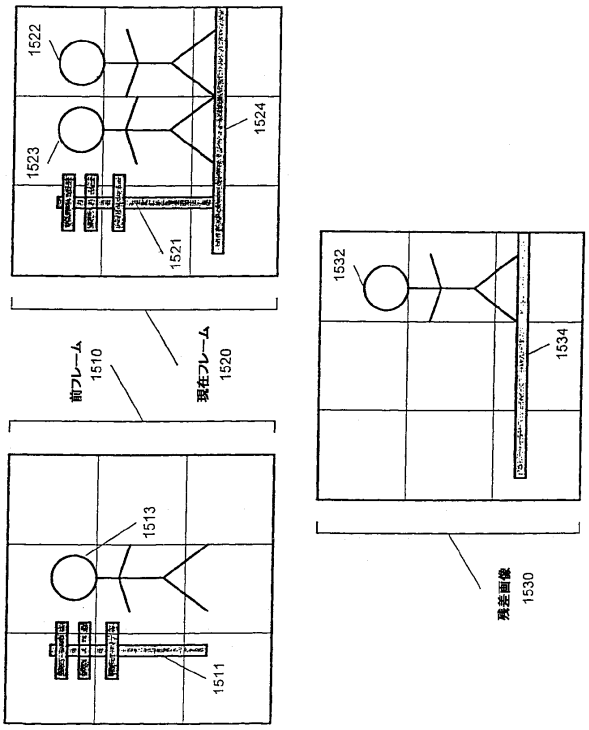
【図１８Ｂ】



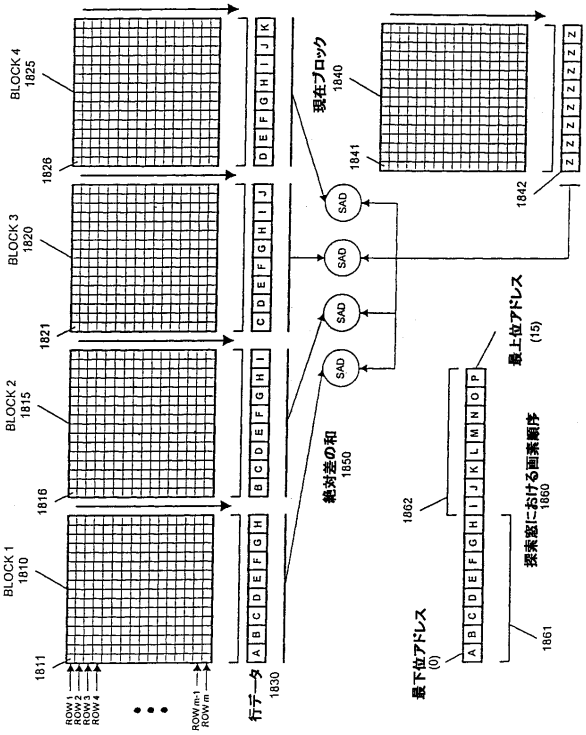
【図１９Ｂ】



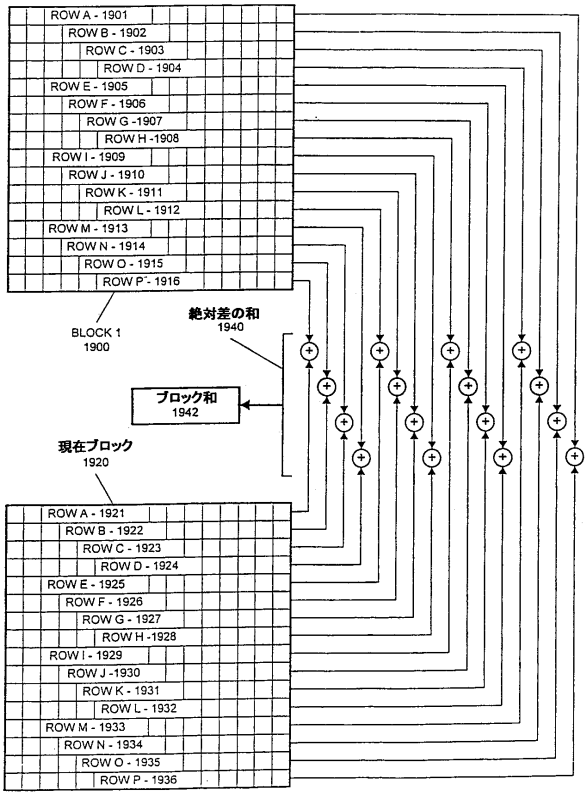
【図２０】



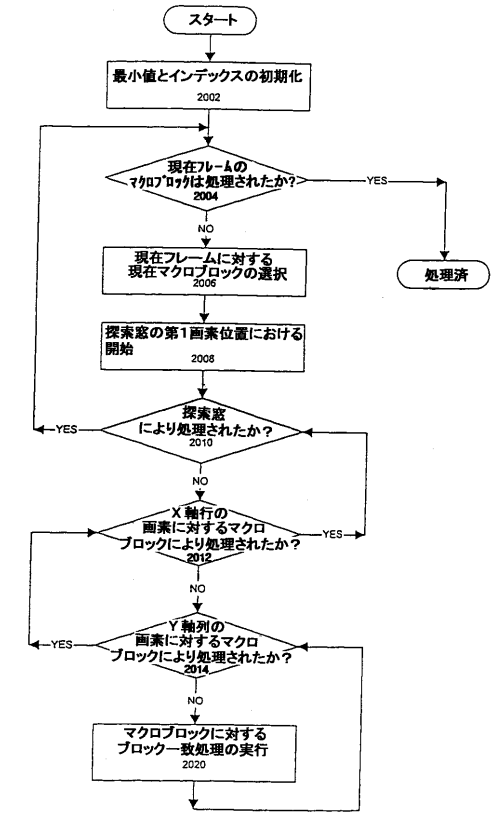
【図２２Ｃ】



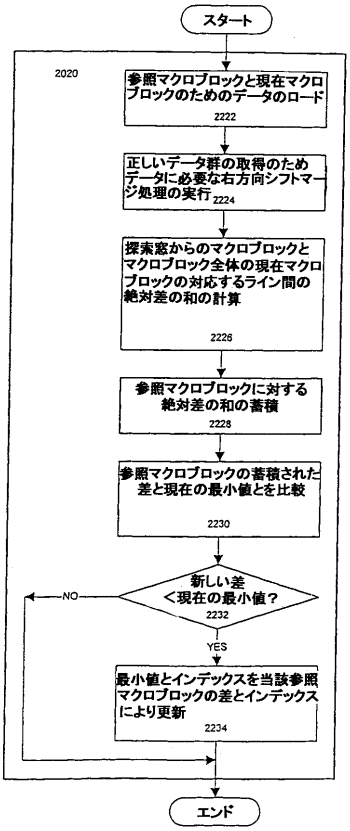
【図２２Ｄ】



【図２３Ａ】



【図２３Ｂ】



【図５】

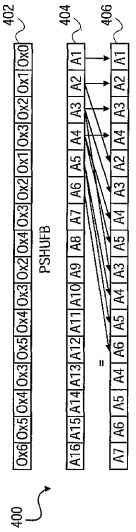


FIG. 5

【図６】

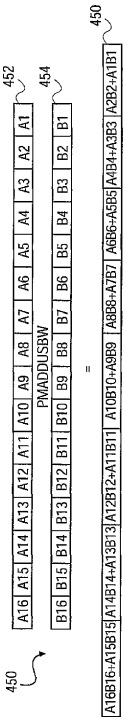


FIG. 6

【図７Ａ】

FIG. 7A

【図７Ｂ】

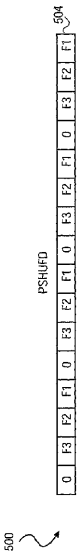


FIG. 7B

【図７Ｃ】

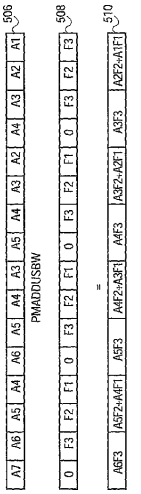


FIG. 7C

【図８Ａ】

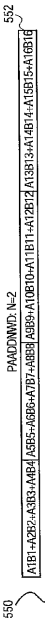


FIG. 8A

【図８Ｂ】

FIG. 8B

【図８Ｃ】

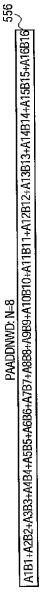


FIG. 8C

【図８Ｄ】

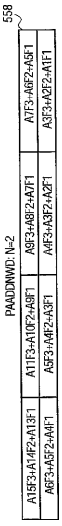


FIG. 8D

【図１７Ａ】

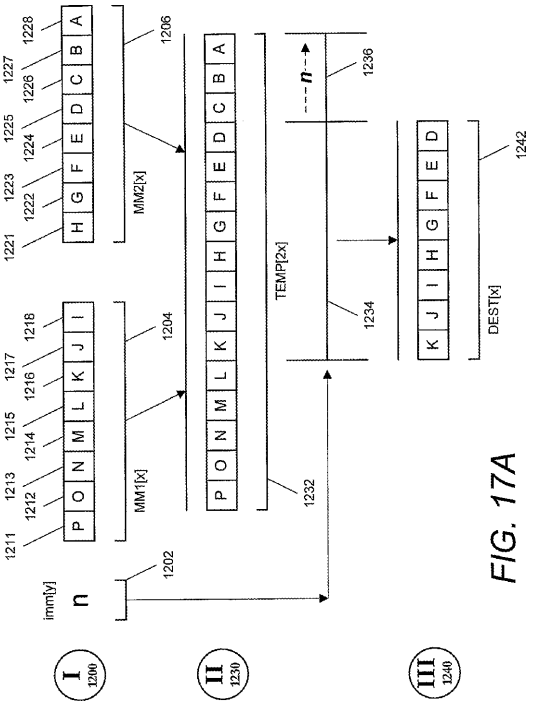


FIG. 17A

【図１７Ｂ】

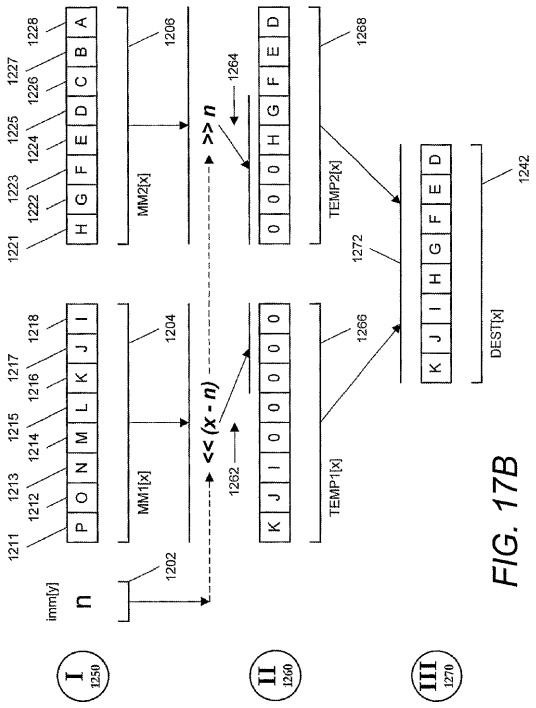


FIG. 17B

【図１９Ａ】

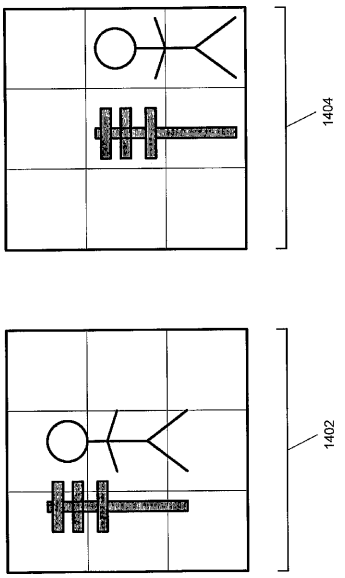


FIG. 19A

【図２１Ａ】

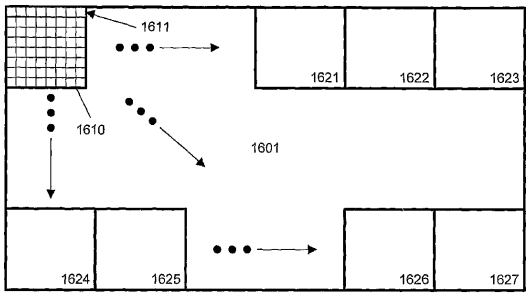


FIG. 21A

【図２１Ｂ】

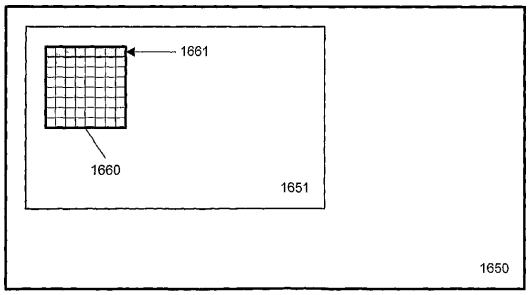


FIG. 21B

【図２２Ａ】

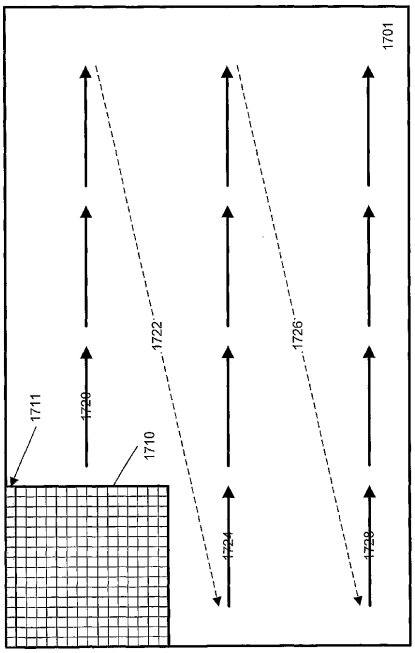


FIG. 22A

【図２２Ｂ】

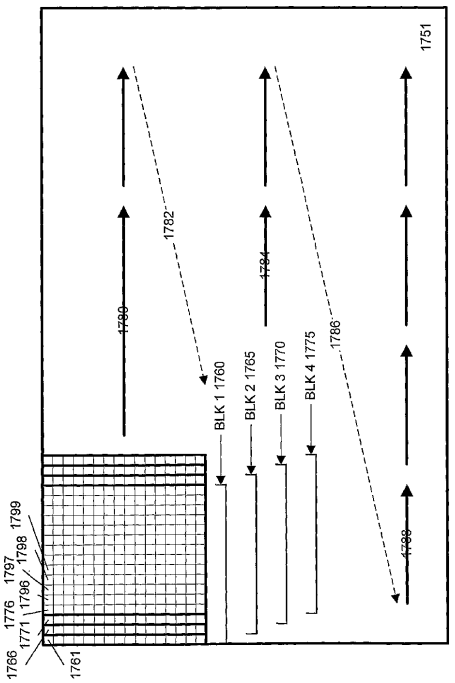


FIG. 22B

フロントページの続き

(72)発明者 セボット，ジュリエン
アメリカ合衆国 ９７１２４ オレゴン州 ヒルズボロ ノースウエスト １８８ス ストリート ２３２３ アパートメント １３２２
(72)発明者 メイシー，ウィリアム，ジュニア
アメリカ合衆国 ９４３０１ カリフォルニア州 パロアルト メルヴィル アヴェニュー １５１
(72)発明者 デベス，エリック
アメリカ合衆国 ９５０５４ カリフォルニア州 サンタクララ レキシントン ストリート １３６５
(72)発明者 ギュエン，ヒュイ
アメリカ合衆国 ７８６６０ テキサス州 フルジャーヴィル アイル オブ マン ロード １６９０９

審査官 緑川 隆

(56)参考文献 Peleg，A. Weiser，U.，MMX technology extension to the Intel architecture，IEEE Micro，米国，IEEE，１９９６年 ８月，Volume: 16， Issue: 4，p.42-50
松原 敦，Pentium IIIの新命令を検証，第３部 新命令セット，プログラミング・テクニックで性能は大きく変わる，日経バイト，日本，日経ＢＰ社 Nikkei Business Publications,Inc.，１９９９年 3月１２日，第189号，p.136-143
SUGATECH，Pentium IIIプログラミング，C MAGAZINE，日本，ソフトバンクパブリッシング株式会社，１９９９年 ４月２３日，第11巻 第5号，p.42-49

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 7/24
G06F 17/10