(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第4480997号
(P4480997)

(45)発行日　平成22年6月16日(2010.6.16)　(24)登録日　平成22年3月26日(2010.3.26)

(51)Int.Cl.			F I		
G06F	7/38	(2006.01)	G06F	7/38	B
G06F	7/496	(2006.01)	G06F	7/496	
G06F	9/305	(2006.01)	G06F	9/30	340A
G06F	9/315	(2006.01)	G06F	9/30	340D

請求項の数 4　(全 32 頁)

(21)出願番号　特願2003-425711 (P2003-425711)
(22)出願日　平成15年12月22日 (2003.12.22)
(65)公開番号　特開2005-25718 (P2005-25718A)
(43)公開日　平成17年1月27日 (2005.1.27)
審査請求日　平成18年12月13日 (2006.12.13)
(31)優先権主張番号　610833
(32)優先日　平成15年6月30日 (2003.6.30)
(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
(72)発明者　ジェイムズ　シー　アベル
アメリカ合衆国　アリゾナ州　８５０４８　フェニックス　イースト　ロックレッジ　ロード　２３１０

最終頁に続く

(54)【発明の名称】ＳＩＭＤ整数乗算上位丸めシフト

(57)【特許請求の範囲】
【請求項１】
乗算上位丸めシフト処理を実行するためのコンピュータにより実現される方法であって、
当該方法は、Ｌ個のデータ要素の第１セットを有する第１レジスタにおける第１オペランドと、Ｌ個のデータ要素の第２セットを有する第２レジスタにおける第２オペランドとを特定する単一命令に応答して、
マイクロプロセッサが、
各ペアが、前記L個のデータ要素の第１セットからの第１データ要素と、前記L個のデータ要素の第２セットの対応するデータ要素位置からの第２データ要素とを有するL個のデータ要素ペアを掛け合わせ、L個の積のセットを生成するステップと、
前記L個の積のそれぞれを右に１４ビットシフトし、L個のシフトされた値を１８ビット長となるように生成するステップと、
前記L個のシフトされた値のそれぞれの最下位ビット位置に“１”を付加することによって、前記L個のシフトされた値のそれぞれを丸め処理し、L個の丸められた値を生成するステップと、
前記L個の丸められた値のそれぞれを右に１ビットだけスケーリングし、L個のスケーリングされた値のセットを生成するステップと、
L個の切り捨てられた値を取得するため、前記L個のスケーリングされた値から最下位の１６ビットを選択することによって、前記L個のスケーリングされた値のそれぞれを切り

捨て処理し、L個の切り捨てられた値を生成するステップと、
前記単一命令の最終結果として、前記L個の切り捨てられた値を前記単一命令により示される宛先レジスタに格納するステップと、
を実行することによって前記単一命令を実行することからなり、
各切り捨て処理された値は、それのデータ要素のペアに対応するデータ要素位置に格納されることを特徴とする方法。
【請求項２】
単一命令を受け付け、該単一命令に応答して、マイクロプロセッサのハードウェア実行ユニットに２つのオペランドに対してPacked乗算上位丸めシフト処理を実行させるステップと、
前記マイクロプロセッサのハードウェア実行ユニットにおいて前記単一命令を実行し、切り捨て処理された結果のセットを生成するステップと、
Packedデータ要素として宛先レジスタに前記切り捨て処理された結果のセットを格納するステップと、
から構成される方法であって、
前記Packed乗算上位丸めシフト処理は、
Packedデータ要素の第１セットの各データ要素と、Packedデータ要素の第２セットの対応するデータ要素とを乗算し、積のセットを生成し、
前記積のセットのそれぞれを右に１４ビットシフトし、その後に丸め処理して、１８ビット長となるように結果のセットを生成し、
前記結果のそれぞれから複数のビットを選択し、切り捨て処理された結果のセットを生成することから構成され、
前記単一命令は、
前記Packed乗算上位丸めシフト処理に関する情報を提供するため、前記Packed乗算上位丸めシフト処理に対する前記切り捨てられた結果のセットが、前記結果のセットの上位ビット又は下位ビットから構成されるか示すオペコードを指定する第１フィールドと、
前記Packedデータ要素の第１セットを有する第１オペランドに対して、第１ソースアドレスを指定する第２フィールドと、
前記Packedデータ要素の第２セットを有する第２オペランドに対して、第２ソースアドレスを指定する第３フィールドと、
から構成されるフォーマットを有することを特徴とする方法。
【請求項３】
単一命令に応答してPacked乗算丸めシフト処理を実行するマイクロプロセッサのハードウェア実行ユニットから構成される装置であって、
前記ハードウェア実行ユニットは、前記単一命令に応答して、
Packedデータ要素の第１セットの各データ要素と、Packedデータ要素の第２セットの対応するデータ要素とを乗算し、積のセットを生成し、
シフトされた値のそれぞれの最下位ビット位置に“１”を付加することによって、前記積のセットのそれぞれを丸め処理し、結果のセットを生成し、
前記結果のセットのそれぞれを右に１４ビットシフトし、１８ビット長となるように結果の中間セットを生成し、
前記結果の中間セットのそれぞれから複数のビットを選択し、切り捨てられた結果のセットを生成し、
最終結果として前記切り捨てられた結果のセットを格納し、
前記単一命令は、
前記Packed乗算丸めシフト処理に関する情報を提供するため、前記Packed乗算上位丸めシフト処理に対する前記切り捨てられた結果のセットが、前記結果のセットの上位ビット又は下位ビットから構成されるか示すオペコードを指定する第１フィールドと、
前記Packedデータ要素の第１セットを有する第１オペランドに対して、第１ソースアドレスを指定する第２フィールドと、

前記Packedデータ要素の第２セットを有する第２オペランドに対して、第２ソースアドレスを指定する第３フィールドと、
から構成されるフォーマットを有することを特徴とする装置。
【請求項４】
第１命令を格納するメモリと、
前記メモリから前記第１命令をフェッチするプロセッサと、
から構成されるシステムであって、
前記プロセッサは、前記第１命令の実行に応答して、
Packedデータ要素の第１セットの各データ要素と、Packedデータ要素の第２セットの対応するデータ要素とを乗算し、積のセットを生成し、
シフトされた値のそれぞれの最下位ビット位置に“１”を付加することによって、前記積のセットのそれぞれを丸め処理し、一時的結果のセットを生成し、
前記一時的結果のセットのそれぞれをスケーリングし、スケーリングされた一時的結果のセットを生成し、
前記スケーリングされた一時的結果のそれぞれから複数のビットを選択し、切り捨て処理された結果のセットを生成し、
最終結果として前記切り捨て処理された結果のセットを格納し、
前記第１命令は、
前記Packed乗算丸めシフト処理に関する情報であって、符号付き整数のPacked乗算丸めシフト処理を示す情報を提供するオペコードであって、前記切り捨てられた結果のセットのそれぞれの上位ビットを選択するためのオペコードを指定する第１フィールドと、
前記Packedデータ要素の第１セットを有する第１オペランドに対して、第１ソースアドレスを指定する第２フィールドと、
前記Packedデータ要素の第２セットを有する第２オペランドに対して、第２ソースアドレスを指定する第３フィールドと、
から構成されるフォーマットを有することを特徴とするシステム。
【発明の詳細な説明】
【技術分野】
【０００１】
本開示は、数学的演算を実行する処理装置、関連ソフトウェア及びソフトウェアシーケンスの技術分野に関する。
【背景技術】
【０００２】
今日の社会において、コンピュータシステムはますます広く普及しつつある。コンピュータ処理能力は、広い範囲の分野における労働者の効率性と生産性を高めてきた。コンピュータの購入及び所持コストが逓減するにつれ、より多くの消費者が最新のより高速なマシーンを利用することが可能になった。さらに、多くの人々がノートブックコンピュータの利用をその利用に関する柔軟性により享受している。モバイルコンピュータにより、ユーザはオフィスから外出したり、あるいは出先でも、データを携帯し作業を行うことが容易にできる。このような場面は、マーケティングスタッフ、企業の役員、学生においてさえもよく見られるものである。
【０００３】
プロセッサ技術の進歩に伴い、先端的なプロセッサを備えたマシーン上で実行するための新たなソフトウェアコードが生成される。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。ここで、プロセッサ内部において実行されている命令及び処理タイプにより生じる可能性のある１つの問題がある。すなわち、あるタイプの処理には処理の複雑さ及び／あるいは必要とされる回路の種類に基づき、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、プロセッサ内部での複雑な処理を実行する方法を最適化するという動機付けが生じる。

【０００４】
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、近年における計算機の性能向上の多く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、さらに高い計算能力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と通信の融合であろう。
【０００５】
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。これらの処理は大きな計算量を要することから、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記憶装置を利用することにより、効率的な実行を行うための高いレベルでのデータ並列処理が提供されている。
【発明の開示】
【発明が解決しようとする課題】
【０００６】
既存のアーキテクチャの多く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、命令スループットの減少を招き、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。
【０００７】
本発明は、このような問題点に鑑み、Ｐａｃｋｅｄ乗算上位丸めシフト処理を実行するための方法、装置、システム及びマシーンによる読み出し可能な媒体を提供することを目的とする。
【課題を解決するための手段】
【０００８】
上記課題を解決するために、本発明による方法は、Ｌ個のデータ要素の第１セットを有する第１オペランドを受信するステップと、Ｌ個のデータ要素の第２セットを有する第２オペランドを受信するステップと、各自がＬ個のデータ要素の前記第１セットからの第１データ要素とＬ個のデータ要素の前記第２セットの対応するデータ要素位置からの第２データ要素とを有するＬ個のデータ要素のペアを掛け合わせ、Ｌ個の積のセットを生成するステップと、前記Ｌ個の積のそれぞれを丸め処理し、Ｌ個の丸められた値を生成するステップと、前記Ｌ個の丸められた値のそれぞれをスケーリングし、Ｌ個のスケーリングされた値を生成するステップと、宛先への格納のため、前記Ｌ個のスケーリングされた値のそれぞれを、それのデータ要素のペアに対応するデータ要素位置に格納されるよう切り捨て処理するステップからなることを特徴とする。
【０００９】
上記課題を解決するために、本発明による方法は、Ｐａｃｋｅｄデータ要素の第１セットの各データ要素とＰａｃｋｅｄデータ要素の第２セットの対応するデータ要素を掛け合わせ、積のセットを生成するステップと、前記積のセットのそれぞれを丸めシフトし、結果のセットを生成するステップと、前記結果のセットのそれぞれから複数のビットを選択し、切り捨てられた結果のセットを生成するステップからなるＰａｃｋｅｄ乗算丸めシフト処理を２つのオペランドに実行する命令を受信するステップと、前記命令を実行し、Ｐａｃｋｅｄデータ要素として宛先レジスタに格納するために前記切り捨てられた結果のセットを生成するステップからなる方法であって、前記命令は、前記Ｐａｃｋｅｄ乗算丸めシフト処理についての情報を提供するオペコードを特定する第１フィールド、前記Ｐａｃｋｅｄデータ要素の第１セットを有する第１オペランドのための第１ソースアドレスを特

定する第２フィールド、前記Ｐａｃｋｅｄデータ要素の第２セットを有する第２オペランドのための第２ソースアドレスを特定する第３フィールドからなるフォーマットを有することを特徴とする。

【００１０】

上記課題を解決するために、本発明による装置は、Ｐａｃｋｅｄ乗算丸めシフト処理を実行するための少なくとも１つの命令を含む命令セットの１つ以上の命令を実行する実行ユニットからなる装置であって、前記実行ユニットは、前記Ｐａｃｋｅｄ乗算丸めシフト処理を実行する前記少なくとも１つの命令に応答して、Ｐａｃｋｅｄデータ要素の第１セットの各データ要素とＰａｃｋｅｄデータ要素の第２セットの対応するデータ要素を掛け合わせ、積のセットを生成し、前記積のセットのそれぞれを丸めシフト処理し、結果のセットを生成し、前記結果のそれぞれから複数のビットを選択し、切り捨てられた結果のセットを生成し、前記少なくとも１つの命令は、前記Ｐａｃｋｅｄ乗算丸めシフト処理についての情報を提供するためのオペコードを特定する第１フィールドと、前記Ｐａｃｋｅｄデータ要素の第１セットを有する第１オペランドのための第１ソースアドレスを特定する第２フィールドと、前記Ｐａｃｋｅｄデータ要素の第２セットを有する第２オペランドのための第２ソースアドレスを特定する第３フィールドからなるフォーマットを有することを特徴とする。

【００１１】

上記課題を解決するために、本発明によるシステムは、データ及び命令を格納するメモリと、バスを介し前記メモリに接続され、乗算丸めシフト命令に応答して、乗算丸めシフト処理を実行するプロセッサからなるシステムであって、前記プロセッサは、前記メモリから前記乗算丸めシフト命令を受信するバスユニットと、前記バスユニットに接続され、前記乗算丸めシフト命令を実行する実行ユニットから構成され、前記乗算丸めシフト命令は前記実行ユニットに、Ｐａｃｋｅｄデータ要素の第１セットの各データ要素とＰａｃｋｅｄデータ要素の第２セットの対応するデータ要素を掛け合わせることにより積のセットを生成させて、前記積のセットのそれぞれを丸めシフト処理することにより結果のセットを生成させて、及び前記結果のそれぞれから複数のビットを選択することにより切り捨てられた結果のセットを生成させることを特徴とする。

【００１２】

上記課題を解決するために、本発明によるマシーン読み出し可能な媒体は、プログラムを格納するマシーン読み出し可能な媒体であって、マシーンにより実行可能な前記プログラムは、Ｌ個のデータ要素の第１セットを有する第１オペランドを受信するステップと、Ｌ個のデータ要素の第２セットを有する第２オペランドを受信するステップと、各自がＬ個のデータ要素の前記第１セットからの第１データ要素とＬ個のデータ要素の前記第２セットの対応するデータ要素位置からの第２データ要素とを有するＬ個のペアを掛け合わせ、Ｌ個の積のセットを生成するステップと、前記Ｌ個の積のそれぞれを丸め処理し、Ｌ個の丸められた値を生成するステップと、前記Ｌ個の丸められた値のそれぞれをスケーリングし、Ｌ個のスケーリングされた値を生成するステップと、宛先への格納のため、前記Ｌ個のスケーリングされた値のそれぞれを、そのデータ要素のペアに対応するデータ要素位置に格納されるよう切り捨て処理するステップからなる方法を実行することを特徴とする。

【発明の効果】

【００１３】

以上のように、本発明によれば、Ｐａｃｋｅｄ乗算上位丸めシフト処理を実行するための方法、装置、システム及びマシーンによる読み出し可能な媒体が得られる。

【発明を実施するための最良の形態】

【００１４】

以下、本発明の実施の形態について図面に基づいて説明する。ここで、本発明は添付される図面に制限されるものではない。また図面中、同一の参照記号は同一の要素を示している。

【００１５】
以下の説明はＳＩＭＤ整数乗算丸めシフト処理の実施例を説明する。以下の説明において、本発明のより完全なる理解を提供するために、プロセッサタイプ、マイクロアーキテクチャ状態、イベント、実施可能な機構などのような具体的詳細が与えられる。しかしながら、本発明はこのような具体的詳細以外でも実践可能であるということは当業者には認識されるであろう。さらに、周知の構成及び回路などは、本発明を不必要に不明瞭にしないよう詳細には示されていない。
【００１６】
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、Ｐａｃｋｅｄデータに対する操作が必要とされ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。
【００１７】
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、単に本発明の一例を提供することを目的としている。
【００１８】
以下の実施例は実行ユニットや論理回路に関する命令処理や配置を説明するが、本発明の他の実施例はソフトウェアにより達成可能である。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が本発明の各ステップを実行する。本発明に従う処理を実行するコンピュータ（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格納したマシーンまたはコンピュータによる読み出し可能な媒体を含む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本発明は提供される。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。このようなソフトウェアはシステム内のメモリに格納することが可能である。同じように、そのようなコードはネットワークを介し、あるいは他のコンピュータ読み出し可能な媒体により配信可能である。
【００１９】
従って、マシーン読み出し可能な媒体は、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、電子、光、音響あるいは他の搬送信号（例えば、搬送波、赤外線信号、デジタル信号など）などのマシーン（例えば、コンピュータ）により読み出し可能な形態での情報の格納及び送信のための任意の機構が含まれる。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。

【００２０】

設計は、制作からシミュレーションそして製造と様々な段階を経ているかもしれない。設計を表すデータは様々な方法で当該設計を表しているかもしれない。まず、シミュレーションにおいて有益なように、ハードウェア記述言語や他の機能記述言語を使ってハードウェアが表現される。さらに、論理及び／あるいはトランジスタゲートによる回路レベルのモデルが設計処理のある段階において生成される。さらに、ある段階では、大部分の設計がハードウェアモデルの様々な装置の物理的配置を表すデータレベルに達する。従来の半導体製造技術が利用される場合、ハードウェアモデルを表すデータは、集積回路の生成に利用されるマスクのマスクレイヤに関する様々な特徴の有無を特定するデータであるかもしれない。任意の設計表現において、このデータは任意の形態のマシーン読み出し可能な媒体に格納することができる。このような情報の送信のため生成あるいは変調される光または電気波、メモリ、ディスクのような磁気または光記憶装置などは、マシーン読み出し可能な媒体である。これらの媒体の何れもが設計やソフトウェア情報を「搬送」または「表現」することができる。コードや設計を示し搬送する電気搬送波が送信されるとき、電気信号のコピー、バッファリングあるいは再送が実行される程度まで新たなコピーが行われる。従って、通信プロバイダやネットワークプロバイダは本発明のテクニックを実現するもの（搬送波）のコピーを行える。

【００２１】

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を高速に実行させることが望ましい。しかしながら、より大きな複雑さを有し、より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。

【００２２】

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。

【００２３】

２の補数表記（ｔｗｏ’ｓ－ｃｏｍｐｌｅｍｅｎｔ　ｎｏｔａｔｉｏｎ）は符号付きの数を表現する効果的な方法である。２の補数の最上位ビットがその符号を表し、残りのビットがその大きさを表す。固定小数点数の計算はオーバーフローを引き起こすことなく整数プロセッサにおける乗算を可能にする。小数計算は、掛け算でのオーバーフローに関する問題がない場合、デジタル信号処理プログラミングにとって大いに有益なものである。

２つの１６ビット数の乗算には結果のため３２ビットが必要であり、２つの１６ビット固定小数点数を乗じることにより生成される３２ビットの結果は最小エラーの導入により１６ビットに丸められる。１６ビット整数の変換は、当該整数の小数値を３２７６８により除することである。一実施例では、２つの小数を掛け合わせることにより生じる積の上位１６ビットが着目される。しかしながら、結果の上位１６ビットが期待される小数の結果の半分である。この積が結果に２を乗じるため左方向にシフトされる必要がある。これにより最終的な正しい積が得られる。小数演算はまた乗数及び被乗数の符号拡張を要する。

【００２４】

左方向へのシフトの必要性はまた、小数位置の配置として説明することができる。例えば、小数を掛け合わせるとき、小数点は無視され、最後に置かれる。この小数点は、乗数と被乗数の小数点の右側の合計桁数がそれらの積の小数点の右側の桁数に等しくなるよう配置される。同様に、小数演算のためのここでの「小数点」は最左（符号）ビットの右に位置し、この点の右には１５ビット（桁）ある。しかしながら、ソースにおける小数点の右側には合計３０ビットある。シフトがなければ、３２ビットの結果における小数点の右側は３１ビットとなるであろう。数を１ビットだけ左にシフトすることにより、小数点の右側のビット数を３０に効果的に減らすことができる。

【００２５】

本発明の実施例は、固定点整数ＳＩＭＤ命令の精度を向上させることができる。固定点整数フォーマットは、固定点小数点数演算のものと類似している。一実施例の「１．１５」の固定点フォーマットは、２進数点（ｂｉｎａｒｙ　ｐｏｉｎｔ）が第１４ビットと第１５ビットの間に位置する符号付きの値を有する数を表す。ここでは、ビット位置は最右ビットから０からカウントされる。従って、最右または最下位ビットは第０ポジションとなる。そのすぐ左のビット位置が第１ビット、そして以下同様となる。この１．Ｎ数値フォーマットはしばしばデジタル信号処理（ＤＳＰ）アプリケーションにおいて利用される。本発明による実施例はまた、丸め処理やシフト処理技術を通じさらなる精度の向上を提供する。本発明の実施例から得られるさらなる精度向上は、多くのアプリケーションのより容易なプログラミングに寄与する。さらに、このさらなる精度向上は、映像及び画像処理アプリケーションにおいてしばしば利用される離散コサイン変換（ＤＣＴ）のようなアルゴリズムのより高速な実行を可能にする。

【００２６】

ＳＩＭＤ整数乗算上位丸めシフト命令（ＳＩＭＤ　ｉｎｔｅｇｅｒ　ｍｕｌｔｉｐｌｙ　ｈｉｇｈ　ｗｉｔｈ　ｒｏｕｎｄ　ａｎｄ　ｓｈｉｆｔ　ｉｎｓｔｒｕｃｔｉｏｎ）のための１例となるアプリケーションは高品質の映像においてである。１６ビットの結果を有する１６×１６ビット乗算は映像エンコーダ及びデコーダ、特に、逆ＤＣＴ、ＤＣＴ、量子化（Ｑ）及び逆Ｑブロックにおいて非常によく利用される。乗算の精度は全体の画質に大きな影響がある。本発明の実施例によるパフォーマンス向上及び高速化は、逆ＤＣＴ計算においてより大きな影響力を有する。ＤＣＴ計算に加え、基本的に１６ビットの乗算であるＱ及び逆Ｑ計算にも有益である。

【００２７】

一般に、コンピュータ産業では、８×８逆離散コサイン変換の実現のためＩＥＥＥ規格１１８０－１９９０がよく利用されている。この規格はテレビ会議に関するものであるが、当該規格の一部は様々なＭＰＥＧフォーマットによるエンコーダ及びデコーダに適用されている。しかしながら、高いパフォーマンスを維持しながらＩＥＥＥ１１８０－１９９０規格に準拠することは困難である。このトレードオフは、しばしば非準拠高パフォーマンスまたは準拠低パフォーマンスとなる。さらに、規格への符号化は、特に適切でないアルゴリズムが選ばれている場合には、時間のかかる繰り返しの処理である。

【００２８】

ＩＥＥＥ１１８０－１９９０規格に準拠は、乗数上位丸めシフト命令の実施例により容易になる。本発明によるＳＩＭＤ整数乗算上位丸めシフト命令の実施例は、Ｐａｃｋｅｄデータ環境での入出力データ要素に同一の１．１５データフォーマットを提供することが

できる。これにより、乗算上位丸めシフト処理の実施例を含む命令セットによるコード記述とプログラミングはより簡単化される。同じように、高レベル言語と関連するコンパイラのアクセス性もまた可能になる。映像、音声及び画像エンコーダ／デコーダ（コーデック）のパフォーマンス及び精度を向上させるために、開発者は整数乗算丸めシフトのような固定点ＳＩＭＤ命令の実施例により可能となる言語及びコンパイラを利用することができる。ＳＩＭＤ機能を備えた命令セットは、類似データの繰り返し処理において以前に必要とされた冗長なアルゴリズムの回避に役立つ。

【００２９】

一実現形態における乗算への各入力は１．１５フォーマットに従う。メモリを備えた乗算上位丸めシフト処理の一実施例において、２．１６フォーマットを有する仮の１８ビット値が、２つの１６ビットデータ値の乗算による３２ビットの積の上位ビットから生成される。この仮の１８ビット値は、最下位ビットに「１」を加えることにより、精度のため丸められる。いくつかのテクニックではすべての下位ビットが単に破棄されるが、本発明の実施例による丸め処理は逆ＤＣＴ符号化のためのある許容可能な閾値にエラーが収まることを可能にする。この丸められた値は、さらなる精度のためと、所望の出力フォーマットを得るために、１ビットだけ左にシフトされる。１．１５フォーマットを有する１６ビットの結果が、丸められたシフトされた１８ビット値から抽出される。仮の値に対し実行された丸め及びシフト処理は、３２ビットの積の上位１６ビットを単にとることにより、さらなる精度を有する２ビットを提供することができる。例えば、ここで説明される一般的な実施例では、丸め処理は、３２ビットの積からの上位１６ビットの抽出により、さらなる精度を有する１ビットを提供する。同じように、シフト処理は、丸められた積に対しさらなる精度を有する１ビットを提供する。これらの説明は１６ビット長の整数値に関し実施例を説明しているが、他の実施例は任意のビット長のデータ値に適用することができる。

【００３０】

図１Ａは、本発明の一実施例による乗算上位丸めシフト処理のための命令を実行する実行ユニットを含むプロセッサにより構成される一例となるコンピュータシステムのブロック図である。システム１００は、ここで説明される実施例のような本発明によるデータ処理アルゴリズムを実行する論理を含む実行ユニットを利用するプロセッサ１０２のような構成要素を備える。システム１００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より入手可能なＰＥＮＴＩＵＭ（登録商標）ＩＩＩ、ＰＥＮＴＩＵＭ（登録商標）４、Ｘｅｏｎ（商標）、Ｉｔａｎｉｕｍ（登録商標）及び／またはＸＳｃａｌｅ（商標）マイクロプロセッサに基づく処理システムにより代表される。しかしながら、（他のマイクロプロセッサ、エンジニアリング・ワークステーション、セットトップボックスなどを含む）他のシステムが利用されてもよい。一実施例では、サンプルシステム１００は、ワシントン州レッドモンドのマイクロソフトコーポレーションから入手可能なあるバージョンのＷＩＮＤＯＷＳ（登録商標）オペレーティングシステムを実行してもよい。しかしながら、（例えば、ＵＮＩＸ（登録商標）やＬｉｎｕｘのような）他のオペレーティングシステム、埋め込みソフトウェア及び／またはグラフィカルユーザインタフェースが使われてもよい。本発明は、ハードウェア回路とソフトウェアの特定の組み合わせに限定されるものではない。

【００３１】

本発明の他の実施例では、携帯装置や埋め込みアプリケーションのような他の装置において利用可能である。携帯装置の例として、携帯電話、インターネットプロトコル装置、デジタルカメラ、ＰＤＡ（Ｐｅｒｓｏｎａｌ　Ｄｉｇｉｔａｌ　Ａｓｓｉｓｔａｎｔ）、及び携帯型パーソナルコンピュータなどが含まれる。埋め込みアプリケーションには、マイクロコントローラ、デジタル信号プロセッサ（ＤＳＰ）、システムオンチップ、ネットワークコンピュータ（ＮｅｔＰＣ）、セットトップボックス、ネットワークハブ、ワイドエリアネットワーク（ＷＡＮ）スイッチ、あるいはずれたメモリコピーまたは移動を実行する他のシステムなどが含まれる。さらに、マルチメディアアプリケーションの効率向上

のため、同時に複数のデータに対し処理する命令を可能にするよう実現されたアーキテクチャがある。データタイプやデータ量が増大するに従い、より効率的な方法によりデータを操作できるようコンピュータ及びそのプロセッサの性能を向上させねばならない。

【００３２】

図１Ａは、本発明による丸め及びシフト命令によるＳＩＭＤ整数乗算上位を含むアルゴリズムを処理する１つ以上の実行ユニット１０８を備えるプロセッサ１０２により構成されるコンピュータシステム１００のブロック図である。例えば、プロセッサ１０２は、Ｐａｃｋｅｄデータオペランドに対するＳＩＭＤ乗算上位処理をリクエストするプログラム命令を受信することができる。本実施例は単一のプロセッサデスクトップまたはサーバシステムに関して説明されるが、他の実施例がマルチプロセッサシステムに含まれてもよい。システム１００はハブアーキテクチャの一例である。コンピュータシステム１００は、データ信号を処理するプロセッサ１０２を備える。プロセッサ１０２は、複合命令セットコンピュータ（ＣＩＳＣ）マイクロプロセッサ、縮小命令セットコンピュータ（ＲＩＳＣ）マイクロプロセッサ、ＶＬＩＷ（Ｖｅｒｙ Ｌｏｎｇ Ｉｎｓｔｒｕｃｔｉｏｎ Ｗｏｒｄ）マクロプロセッサ、命令セットの組み合わせを実現するプロセッサ、あるいはデジタル信号プロセッサのような他のプロセッサ装置でありうる。プロセッサ１０２は、プロセッサ１０２とシステム１００内の他の構成要素との間のデータ信号を送信することができるプロセッサバス１１０に接続される。システム１００の構成要素は、当業者には周知のそれの既存の機能を実行する。

【００３３】

一実施例では、プロセッサ１０２はレベル１（Ｌ１）内部キャッシュメモリ１０４を含む。アーキテクチャに応じて、プロセッサ１０２は単一の内部キャッシュあるいは複数レベルの内部キャッシュを有する。また他の実施例では、キャッシュメモリはプロセッサ１０２の外部に設けられていてもよい。他の実施例ではまた、要求される実施形態に応じて、内部キャッシュと外部キャッシュの両方の組み合わせが含まれうる。レジスタファイル１０６は、様々なタイプのデータを整数レジスタ、浮動小数点レジスタ、ステータスレジスタ及び命令ポインタレジスタを含む様々なレジスタに格納することができる。

【００３４】

実行ユニット１０８は、整数及び浮動小数点処理を実行する論理を含み、プロセッサ１０２に設けられる。プロセッサ１０２はまた、あるマクロ命令のためのマイクロコードを格納するマイクロコード（ｕｃｏｄｅ）ＲＯＭを備えていてもよい。本実施例では、実行ユニット１０８はＰａｃｋｅｄ命令セット１０９を扱う論理を含んでいる。一実施例において、Ｐａｃｋｅｄ命令セット１０９は、結果として得られる積の関連する上位部分を獲得するため、Ｐａｃｋｅｄ乗算上位命令を含んでいる。汎用プロセッサ１０２の命令セットのＰａｃｋｅｄ命令セット１０９を、命令を実行する関連回路と共に含めることにより、多くのマルチメディアアプリケーションにより利用される処理が、汎用プロセッサ１０２のＰａｃｋｅｄデータを使うことにより実行されてもよい。これにより、Ｐａｃｋｅｄデータに処理を実行するためのプロセッサのデータバスの全幅を使用することにより、多くのマルチメディアアプリケーションがより効率的に実行されうる。この結果、１つのデータ要素に１つ以上の処理を同時に実行するために、より小さいデータユニットをプロセッサのデータバスに送信する必要がなくなる。実行ユニット１０８の他の実施例はまた、マイクロコントローラ、埋め込みプロセッサ、グラフィックス装置、ＤＳＰ及び他のタイプの論理回路において利用可能である。システム１００はメモリ１２０を備える。メモリ１２０は、ＤＲＡＭ（Ｄｙｎａｍｉｃ Ｒａｎｄｏｍ Ａｃｃｅｓｓ Ｍｅｍｏｒｙ）装置、ＳＲＡＭ（Ｓｔａｔｉｃ Ｒａｎｄｏｍ Ａｃｃｅｓｓ Ｍｅｍｏｒｙ）装置、フラッシュメモリ装置、あるいは他のメモリ装置であってもよい。メモリ１２０は、プロセッサ１０２により実行可能なデータ信号により表される命令及び／あるいはデータを格納することができる。

【００３５】

システム論理チップ１１６は、プロセッサバス１１０とメモリ１２０に接続される。例

示された実施例のシステム論理チップ１１６はメモリコントローラハブ（ＭＣＨ）である。プロセッサ１０２は、プロセッサバス１１０を介しＭＣＨ１１６と通信することができる。ＭＣＨ１１６は、命令及びデータの格納と、グラフィックスコマンド、データ及びテクスチャの格納のため、メモリ１２０への高帯域幅メモリパス１１８を与える。ＭＣＨ１１６は、プロセッサ１０２、メモリ１２０及びシステム１００の他の構成要素間においてデータ信号を導き、プロセッサバス１１０、メモリ１２０及びシステムＩ／Ｏ１２２間においてデータ信号をブリッジする。いくつかの実施例では、システム論理チップ１１６は、グラフィックスコントローラ１１２への接続のために、グラフィックスポートを備えうる。ＭＣＨ１１６は、メモリインタフェース１１８を介しメモリ１２０に接続される。グラフィックスカード１１２は、ＡＧＰ（Ａｃｃｅｌｅｒａｔｅｄ Ｇｒａｐｈｉｃｓ Ｐｏｒｔ）インターコネクト１１４を介しＭＣＨ１１６に接続される。

【００３６】

システム１００は、専用ハブインタフェースバス１２２を使って、ＭＣＨ１１６をＩ／Ｏコントローラハブ（ＩＣＨ）１３０に接続する。ＩＣＨ１３０は、ローカルＩ／Ｏバスを介しいくつかのＩ／Ｏ装置への直接の接続を提供する。ローカルＩ／Ｏバスは、周辺装置をメモリ１２０、チップセット及びプロセッサ１０２に接続する高速Ｉ／Ｏバスである。いくつかの例は、音声コントローラ、ファームウェアハブ（フラッシュＢＩＯＳ）１２８、無線送信機１２６、データ記憶装置１２４、ユーザ入力及びキーボードインタフェースを含む既存のＩ／Ｏコントローラ、ＵＳＢ（Ｕｎｉｖｅｒｓａｌ Ｓｅｒｉａｌ Ｂｕｓ）のようなシリアル拡張ポート、及びネットワークコントローラ１３４である。データ記憶装置１２４は、ハードディスクドライブ、フロッピーディスク（登録商標）ドライブ、ＣＤ－ＲＯＭ装置、フラッシュメモリ装置、または他の大容量記憶装置から構成されうる。

【００３７】

システムの他の実施例では、Ｐａｃｋｅｄ乗算上位命令を実行する実行ユニットはシステムオンチップと共に利用することができる。システムオンチップの一実施例は、プロセッサとメモリから構成される。そのようなシステムのためのメモリはフラッシュメモリである。フラッシュメモリは、プロセッサ及び他のシステム構成要素と同じチップ上に設けられる。さらに、メモリコントローラやグラフィックスコントローラのような他の論理ブロックがまた、システムオンチップ上に配置されうる。

【００３８】

図１Ｂは、本発明の原理を実現するデータ処理システム１４０の他の実施例を示す。データ処理システム１４０の一実施例は、（「ｗｗｗ．ｉｎｔｅｌ．ｃｏｍ」にて説明されるような）インテルＸＳｃａｌｅ（商標）技術によるインテル（登録商標）パーソナルインターネットクライアントアーキテクチャ（ＰＣＡ）アプリケーションプロセッサである。ここで説明される実施例は、発明の範囲から逸脱することなく他の処理システムと共に利用することができるということは、当業者には認識されるであろう。

【００３９】

コンピュータシステム１４０は、乗算上位丸めシフトを含むＳＩＭＤ処理を実行することができる処理コア１５９を備える。一実施例において、処理コア１５９は、ＣＩＳＣ、ＲＩＳＣ、ＶＬＩＷタイプアーキテクチャに限定されることなく任意のタイプのアーキテクチャの処理ユニットを表す。処理コア１５９はまた、１以上の処理技術における製造に適したものであってもよいし、十分詳細にマシーン読み出し可能なメディアに表されることにより、処理コア１５９はこの製造の容易化に適したものであってもよい。

【００４０】

処理コア１５９は、実行ユニット１４２、レジスタファイルセット１４５、及びデコーダ１４４から構成される。処理コア１５９はまた、本発明の理解に必要でない追加的な回路（図示せず）を含んでもよい。実行ユニット１４２は、処理コア１５９により受信された命令の実行に利用される。典型的なプロセッサ命令の認識に加えて、実行ユニット１４２はＰａｃｋｅｄデータフォーマットに対する処理の実行のため、Ｐａｃｋｅｄ命令セッ

ト１４３における命令を認識することができる。Ｐａｃｋｅｄ命令セット１４３は、データマージ処理をサポートする命令を含み、またさらに他のＰａｃｋｅｄ命令を含んでいてもよい。実行ユニット１４２は内部バスによりレジスタファイル１４５に接続される。レジスタファイル１４５は、データを含む情報の格納のための処理コア１５９における記憶領域を表す。前述のように、Ｐａｃｋｅｄデータの格納に利用される記憶領域が何れであるかは重要ではないということは理解されるであろう。実行ユニット１４２はデコーダ１４４に接続される。デコーダ１４４は、処理コア１５９により受信された命令を制御信号及び／あるいはマイクロコード入力ポイントに復号するために利用される。これらの制御信号及び／あるいはマイクロコード入力ポイントに応答して、実行ユニット１４２は適切な処理を実行する。

【００４１】

処理コア１５９は、以下に限定されるものではないが、例えば、ＳＤＲＡＭ（Ｓｙｎｃｈｒｏｎｏｕｓ Ｄｙｎａｍｉｃ Ｒａｎｄｏｍ Ａｃｃｅｓｓ Ｍｅｍｏｒｙ）コントロール１４６、ＳＲＡＭ（Ｓｔａｔｉｃ Ｒａｎｄｏｍ Ａｃｃｅｓｓ Ｍｅｍｏｒｙ）コントロール１４７、バーストフラッシュメモリインタフェース１４８、ＰＣＭＣＩＡ（Ｐｅｒｓｏｎａｌ Ｃｏｍｐｕｔｅｒ Ｍｅｍｏｒｙ Ｃａｒｄ Ｉｎｔｅｒｎａｔｉｏｎａｌ Ａｓｓｏｃｉａｔｉｏｎ）／ＣＦ（Ｃｏｍｐａｃｔ Ｆｌａｓｈ）カードコントロール１４９、液晶（ＬＣＤ）コントロール１５０、ＤＭＡ（Ｄｉｒｅｃｔ Ｍｅｍｏｒｙ Ａｃｃｅｓｓ）コントローラ１５１、及び代替バスマスタインタフェース１５２を含む他の様々なシステム装置と通信するためのバス１４１に接続される。一実施例では、データ処理システム１４０はまた、Ｉ／Ｏバス１５３を介し様々なＩ／Ｏ装置と通信するためのＩ／Ｏブリッジ１５４を備える。このようなＩ／Ｏ装置は、以下に限定されるものではないが、例えば、ＵＡＲＴ（Ｕｎｉｖｅｒｓａｌ Ａｓｙｎｃｈｒｏｎｏｕｓ Ｒｅｃｅｉｖｅｒ／Ｔｒａｎｓｍｉｔｔｅｒ）１５５、ＵＳＢ１５６、ブルートゥース無線ＵＡＲＴ１５７、及びＩ／Ｏ拡張インタフェース１５８から構成されてもよい。

【００４２】

データ処理システム１４０の一実施例は、モバイル、ネットワーク及び／あるいは無線通信のために、シフトマージ処理を含むＳＩＭＤ処理を実行することができる処理コア１５９を備える。処理コア１５９は、ウォルシュアダマール変換、高速フーリエ変換（ＦＦＴ）、離散コサイン変換（ＤＣＴ）及びそれら各自の逆変換などの離散変換と、色空間変換、映像符号化動き予測または映像復号化動き予測などの圧縮／解凍技術と、パルス符号変調（ＰＣＭ）のような変調／復調（ＭＯＤＥＭ）機能とを含む様々な音声、映像、画像形成及び通信アルゴリズムによりプログラムされてもよい。

【００４３】

図１Ｃは、ＳＩＭＤ乗算上位処理を実行することができるデータ処理システムの他の実施例を示す。他の実施例によると、データ処理システム１６０は、メインプロセッサ１６６、ＳＩＭＤコプロセッサ１６１、キャッシュメモリ１６７及び入出力システム１６８を備える。入出力システム１６８は、選択的に無線インタフェース１６９に接続されてもよい。ＳＩＭＤコプロセッサ１６１は、乗算上位を含むＳＩＭＤ処理を実行することができる。処理コア１７０は、１つ以上の処理技術における製造に適したものであってよいし、十分詳細にマシーン読み出し可能なメディアに表すことにより、処理コア１７０はそれを含んだデータ処理システム１６０のすべてあるいは一部の製造を容易化するのに適したものであってもよい。

【００４４】

一実施例において、ＳＩＭＤコプロセッサ１６１は、実行ユニット１６２とレジスタファイルセット１６４から構成される。メインプロセッサ１６５の一実施例は、実行ユニット１６２による実行のため、ＳＩＭＤＰａｃｋｅｄ乗算上位命令を含む命令セット１６３の命令を認識するデコーダ１６５を備える。他の実施例では、ＳＩＭＤコプロセッサ１６１はまた、命令セット１６３の命令を復号するデコーダ１６５Ｂの少なくとも一部を備える。処理コア１７０はまた、本発明の理解に必要でない追加的回路（図示せず）を含む。

【００４５】
動作中、メインプロセッサ１６６は、キャッシュメモリ１６７と入出力システム１６８との相互作用を含む一般的タイプのデータ処理動作を制御するデータ処理命令のストリームを実行する。ＳＩＭＤコプロセッサ命令は、データ処理命令のストリームに埋め込まれる。メインプロセッサ１６６のデコーダ１６５は、装着されたＳＩＭＤコプロセッサ１６１により実行されるべきタイプとしてこれらのＳＩＭＤコプロセッサ命令を発する。従って、メインプロセッサ１６６は、これらのＳＩＭＤコプロセッサ命令（あるいはＳＩＭＤコプロセッサ命令を表す制御信号）をコプロセッサバス１６６において発行し、そこから装着されたＳＩＭＤコプロセッサにより受信される。この場合、ＳＩＭＤコプロセッサ１６１は受信したＳＩＭＤコプロセッサ命令を受領及び実行する。

【００４６】
ＳＩＭＤコプロセッサ命令による処理のため、データは無線インタフェース１６９を介し受信されてもよい。一例として、音声通信はデジタル信号の形式で受け取られ、当該音声信号を表すデジタル音声サンプルを再生成するためＳＩＭＤコプロセッサ命令により処理されてもよい。他の例として、圧縮された音声及び／または映像がデジタルビットストリーム形式で受信され、デジタル音声サンプル及び／または動き映像フレームを再生成するためＳＩＭＤコプロセッサ命令により処理されてもよい。一実施例において、処理コア１７０、メインプロセッサ１６６及びＳＩＭＤコプロセッサ１６１は、実行ユニット１６２、レジスタファイルセット１６４及びデコーダ１６５からなる単一の処理コア１７０に一体化され、ＳＩＭＤ乗算上位命令を含む命令セット１６３の命令を認識する。

【００４７】
図２は、本発明によるＰａｃｋｅｄ整数乗算上位丸めシフト処理を実行する論理回路を有する一実施例のプロセッサ２００のためのマイクロアーキテクチャのブロック図である。丸め及びシフト処理によるＳＩＭＤ整数乗算上位処理はまた、Ｐａｃｋｅｄ乗算上位丸めシフト処理（ＰＭＵＬ上位）、または乗算上位処理と呼ばれるかもしれない。Ｐａｃｋｅｄ乗算上位命令の一実施例において、当該命令により、２つのメモリブロックからデータが抽出され、仮の結果を得るために各ブロックから対応するデータ要素を掛け合わせ、この仮の結果を丸めシフトし、結果として得るマージされたデータブロックにおける格納のため、この中間結果を各積の所望の上位部分に切り捨てる。ＳＩＭＤ乗算上位命令はまた、ＰＭＵＬＨＲＳＷあるいはＰａｃｋｅｄ乗算上位丸めシフトと呼ばれる。本実施例では、マージ処理はまた、バイト、ワード、ダブルワード、クアドワードなどのサイズを有するデータ要素での処理を行うために実行される。ここでの説明は整数及び整数処理に関するものであるが、本発明の他の実施例は浮動小数点数及び浮動小数点処理で利用されてもよい。

【００４８】
イン・オーダーフロントエンド（ｉｎ－ｏｒｄｅｒ　ｆｒｏｎｔ　ｅｎｄ）２０１は、実行対象のマクロ命令をフェッチし、プロセッサパイプラインにおける後の利用のため当該命令を用意するプロセッサ２００の一部である。本実施例のフロントエンド２０１は複数のユニットを備えている。命令プリフェッチャ２２６は、メモリからマクロ命令をフェッチし、命令デコーダ２２８に供給し、マシーンにより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あるいは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２３０は、復号化されたｕｏｐを受け取り、実行のためｕｏｐキュー２３４においてそれらが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２３０が複雑なマクロ命令に直面すると、マイクロコードＲＯＭ２３２が当該処理の完了に必要なｕｏｐを提供する。

【００４９】
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。一実施例では、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２２８はマイクロコードＲＯＭ２３２にアクセスし、マクロ命令を実行する。一実施例では、乗算上位丸めシフト命令が、命

令デコーダ２２８における処理のため少数のマイクロｏｐに復号される。他の実施例では、Ｐａｃｋｅｄ乗算上位丸めシフトアルゴリズムのための命令が、処理の完了に多数のマイクロｏｐが必要とされる場合、マイクロＲＯＭ２３２に格納される。トレースキャッシュ２３０は、入力ポイントＰＬＡ（Ｐｒｏｇｒａｍｍａｂｌｅ Ｌｏｇｉｃ Ａｒｒａｙ）を参照し、マイクロコードＲＯＭ２３２におけるマージ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ＲＯＭ２３２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２０１はトレースキャッシュ２３０からマイクロｏｐの取り込みを再開する。

【００５０】

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２２８が複雑なマクロ命令に直面すると、マイクロコードＲＯＭ２３２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（ｏｕｔ－ｏｆ－ｏｒｄｅｒ ｅｘｅｃｕｔｉｏｎ ｅｎｇｉｎｅ）２０３に通信される。

【００５１】

アウト・オブ・オーダー実行エンジン２０３では、実行のための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。割り当てまたはアロケータ論理は、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、レジスタファイルの入力の論理レジスタを改名する。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２０２、低速／通常浮動小数点スケジューラ２０４、及びシンプル浮動小数点スケジューラ２０６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つへの各ｕｏｐの入力を割り当てる。ｕｏｐスケジューラ２０２、２０４及び２０６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２０２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。

【００５２】

レジスタファイル２０８と２１０は、スケジューラ２０２、２０４及び２０６と、実行ブロック２１１の実行ユニット２１２、２１４、２１６、２１８、２２０、２２２及び２２４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２０８と２１０がある。本実施例のレジスタファイル２０８と２１０のそれぞれはまた、まだレジスタファイルに書き込まれていない終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２０８と浮動小数点レジスタファイル２１０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２０８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２１０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。

【００５３】

実行ブロック２１１は、命令を実際に実行する実行ユニット２１２、２１４、２１６、２１８、２２０、２２２及び２２４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２０８と２１０を含む。本実施例のプロセッサ２００は、アドレス生成ユニット（ＡＧＵ）２１２、ＡＧＵ２１４、高速ＡＬＵ２１６、高速ＡＬＵ２１８、低速ＡＬＵ２２０、浮動小数点ＡＬＵ２２２及び浮動小数点移動ユニット２２４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２２２と２２４は、浮動小数点処理、ＭＭＸ処理、ＳＩＭＤ処理及びＳＳＥ処理を実行する。本実施例の浮動小数点ＡＬＵ２２２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で行われる。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与する。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行われる。

【００５４】

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２１６と２１８において処理される。本実施例の高速ＡＬＵ２１６と２１８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ＡＬＵ２２０に渡される。低速ＡＬＵ２２０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、ＡＧＵ２１２と２１４により実行される。本実施例では、整数ＡＬＵ２１６、２１８及び２２０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、ＡＬＵ２１６、２１８及び２２０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２２２と２２４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２２２と２２４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。

【００５５】

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「レジスタ」という単語が使われる。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、一実施例のレジスタは特定タイプの回路に限定されない。むしろ一実施例のレジスタはデータの格納及び提供、及びここで説明される機能の実行が可能であればよい。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することによる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。一実施例では、整数レジスタは３２ビット整数データを格納している。一実施例のレジスタファイルはまた、Ｐａｃｋｅｄデータのための８つのマルチメディアＳＩＭＤレジスタを含んでいる。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。本実施例では、Ｐａｃｋｅｄデータと整数データの格納において、レジスタは２つのデータタイプ間での区別をする必要はない。

【００５６】

図３Ａは、本発明の一実施例による１２８ビット幅のマルチメディアレジスタでの様々な符号付き及び符合なしＰａｃｋｅｄデータタイプ表現を示す。本実施例のＰａｃｋｅｄ

バイトフォーマットは６つのＰａｃｋｅｄバイトデータ要素を含んでいる。バイトは８ビットデータとして定義される。符号なしＰａｃｋｅｄバイト表現３０２は、ＳＩＭＤレジスタへの符号なしＰａｃｋｅｄバイトの格納を示している。各バイトデータ要素の情報が、第０バイトに対しては第７ビットから第０ビットに、第１バイトに対しては第１５ビットから第８ビットに、第２バイトに対しては第２３ビットから第１６ビットに、最後に第１５バイトに対しては第１２８ビットから第１２０ビットにそれぞれ格納される。従って、利用可能なすべてのビットがレジスタにおいて利用される。この格納配置により、プロセッサの記憶効率の向上がもたらされる。１６のデータ要素がアクセスされると、１つの処理が１６のデータ要素に対し並列に実行される。

【００５７】

符号付きＰａｃｋｅｄバイト表現３０４は符号付きＰａｃｋｅｄバイトの格納を示す。すべてのバイトデータ要素の第８ビットは符号標識である。本実施例のＰａｃｋｅｄワードフォーマットは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。符号なしＰａｃｋｅｄワード表現３０６は、第７ワードから第９ワードがＳＩＭＤレジスタにどのように格納されているかを示している。符号付きＰａｃｋｅｄワード表現３０８は符号なしＰａｃｋｅｄワードイン・レジスタ表現３０６と同様である。ここで、各ワードデータ要素の第１６ビットは符号標識である。Ｐａｃｋｅｄダブルワードフォーマットは１２８ビット長であり、４つのＰａｃｋｅｄダブルワードデータ要素を含んでいる。各Ｐａｃｋｅｄダブルワード要素は３０ビットの情報を含んでいる。符号なしＰａｃｋｅｄダブルワード表現３１０は、ダブルワード要素がどのように格納されているかを示している。符号付きＰａｃｋｅｄダブルワード表現３１２は符号なしＰａｃｋｅｄダブルワードイン・レジスタ表現３１０と同様である。ここで、必要な符号ビットは各ダブルワードデータ要素の第３２ビットである。Ｐａｃｋｅｄクアドワードは１２８ビット長であり、２つのＰａｃｋｅｄクアドワードデータ要素を含んでいる。

【００５８】

一般に、データ要素は、１つのレジスタあるいはメモリ領域に同じ長さの他のデータ要素と共に格納されるデータ部分である。ＳＳＥ２技術に関するＰａｃｋｅｄデータシーケンスにおいて、ＸＭＭレジスタに格納されるデータ要素数は、各データ要素のビット長により割られた１２８ビットである。同様に、ＭＭＸ及びＳＳＥ技術に関するＰａｃｋｅｄデータシーケンスでは、ＭＭＸレジスタに格納されるデータ要素数は、各データ要素のビット長により割られた６４ビットである。図３Ａに示されたデータタイプは１２８ビット長であるが、本発明の実施例はまた６４ビット幅あるいは他のサイズのオペランドにおいて動作可能である。

【００５９】

図３Ｂは、他のイン・レジスタデータ記憶フォーマットを示す。各Ｐａｃｋｅｄデータは複数の独立データ要素を含みうる。Ｐａｃｋｅｄハーフ３４１、Ｐａｃｋｅｄシングル３４２及びＰａｃｋｅｄダブル３４３の３つのＰａｃｋｅｄデータフォーマットが示される。Ｐａｃｋｅｄハーフ３４１、Ｐａｃｋｅｄシングル３４２及びＰａｃｋｅｄダブル３４３の一実施例は定点データ要素を含んでいる。他の実施例では、Ｐａｃｋｅｄハーフ３４１、Ｐａｃｋｅｄシングル３４２及びＰａｃｋｅｄダブル３４３の１つ以上が浮動小数点データ要素を含みうる。Ｐａｃｋｅｄハーフ３４１の他の実施例は、８つの１６ビットデータ要素を含む１２８ビット長である。Ｐａｃｋｅｄシングル３４２の一実施例は、１２８ビット長であり、４つの３２ビットデータ要素を含む。Ｐａｃｋｅｄダブル３４３の一実施例は、１２８ビット長であり、２つの６４ビットデータ要素を含む。このようなＰａｃｋｅｄデータフォーマットは、例えば、９６ビット、１６０ビット、１９２ビット、２２４ビット、２５６ビットあるいはそれ以上の他のレジスタ長にさらに拡張することが可能であるということは理解されるであろう。

【００６０】

図３Ｃは、「ｗｗｗ．ｉｎｔｅｌ．ｃｏｍ／ｄｅｓｉｇｎ／ｌｉｔｃｅｎｔｒ」を介し

インテルコーポレーションから利用可能な「ＩＡ－３２インテルアーキテクチャソフトウェア開発者のためのマニュアル２」において説明されている一タイプの処理符号化フォーマット（オペコード）に対応する３２以上のビットを有する処理符号化フォーマットとレジスタ／メモリオペランドアドレッシングモードの一実施例を示す。丸めシフトによる乗算上位処理のタイプが、１つ以上のフィールド３６１と３６２により符号化される。２つまでのソースオペランド識別子３６４と３６５を含めて命令あたり２つまでのオペランドの位置が特定される。シフトマージ命令の一実施例において、宛先オペランド識別子３６６はソースオペランド識別子３６４と同一である。他の実施例においては、宛先オペランド識別子３６６はソースオペランド識別子３６５と同一である。従って、シフトマージ処理の実施例において、ソースオペランド識別子３６４と３６５により特定されるソースオペランドの１つが乗算上位丸めシフト処理の結果により上書きされる。シフトマージ命令の一実施例において、オペランド識別子３６４と３６５は、６４ビットソース及び宛先オペランドの特定に利用されうる。

【００６１】

図３Ｄは、４０ビット以上を有する他の処理符号化（オペコード）フォーマット３７０を示す。オペコードフォーマット３７０は、オペコードフォーマット３６０に対応し、選択的なプレフィックスバイト（ｐｒｅｆｉｘ　ｂｙｔｅ）３７８から構成される。乗算上位丸めシフト処理のタイプが１つ以上のフィールド３７８、３７１及び３７２により符号化される。命令あたり２つまでのオペランド位置がソースオペランド識別子３７４と３７５、及びプレフィックスバイト３７８により特定される。Ｐａｃｋｅｄ乗算上位丸めシフトの一実施例において、１２８ビットソース及び宛先オペランドの特定にプレフィックスバイト３７８が利用される。乗算上位命令の一実施例において、宛先オペランド識別子３７６はソースオペランド識別子３７４と同じである。他の実施例において、宛先オペランド識別子３７６はソースオペランド識別子３７５と同じである。従って、乗算上位処理の実施例において、ソースオペランド識別子３７４と３７５により特定されるソースオペランドの１つが乗算上位処理の結果により上書きされる。オペコードフォーマット３６０と３７０はＭＯＤフィールド３６３と３７３及び選択的なスケール－インデックス－ベース（ｓｃａｌｅ－ｉｎｄｅｘ－ｂａｓｅ）及びディスプレースメント（ｄｉｓｐｌａｃｅｍｅｎｔ）バイトにより部分的に特定されるレジスタ・ツー・レジスタ（ｒｅｇｉｓｔｅｒ　ｔｏ　ｒｅｇｉｓｔｅｒ）、メモリ・ツー・レジスタ（ｍｅｍｏｒｙ　ｔｏ　ｒｅｇｉｓｔｅｒ）、レジスタ・バイ・メモリ（ｒｅｇｉｓｔｅｒ　ｂｙ　ｍｅｍｏｒｙ）、レジスタ・バイ・レジスタ（ｒｅｇｉｓｔｅｒ　ｂｙ　ｒｅｇｉｓｔｅｒ）、レジスタ・バイ・即値（ｒｅｇｉｓｔｅｒ　ｂｙ　ｉｍｍｅｄｉａｔｅ）、レジスタ・ツー・メモリ（ｒｅｇｉｓｔｅｒ　ｔｏ　ｍｅｍｏｒｙ）アドレッシングを可能にする。

【００６２】

図３Ｅに示されるように、他の実施例では、６４ビット単一命令多重データ（ＳＩＭＤ）算術処理が、コプロセッサデータ処理（ＣＤＰ）命令を通じ実行される。処理符号化（オペコード）フォーマット３８０は、ＣＤＰオペコードフィールド３８２と３８９を有するＣＤＰ命令を示す。乗算上位丸めシフト処理の他の実施例では、ＣＤＰ命令のタイプは１つ以上のフィールド３８３、３８４、３８７及び３８８により符号化される。２つまでのソースオペランド識別子３８５と３９０及び１つの宛先オペランド識別子３８６を含めて命令あたり３つまでのオペランド位置が特定される。コプロセッサの一実施例は、８、１６、３２及び６４ビット値に対し動作することができる。一実施例において、固定点または整数データ要素に対し乗算上位処理が実行される。いくつかの実施例では、マージ命令が条件フィールド３８１を利用して条件付きで実行されてもよい。いくつかの乗算上位命令では、ソースデータのサイズがフィールド３８３により符号化される。シフトマージ命令のいくつかの実施例では、ゼロ（Ｚ）、ネガティブ（Ｎ）、キャリー（Ｃ）及びオーバーフロー（Ｖ）の検出がＳＩＭＤフィールドにおいて実行される。いくつかの命令では、サチュレーション（ｓａｔｕｒａｔｉｏｎ）のタイプがフィールド３８４により符号化される。

【００６３】
本発明の一実施形態では、Ｐａｃｋｅｄ乗算上位丸めシフトは、命令フォーマットＰＭＵＬＨＲＳＷmm１、ｍｍ２／ｍ６４により表される。本例におけるＰＭＵＬＨＲＳＷは、Ｐａｃｋｅｄ乗算上位丸めシフトワードの記憶の助けとなるものである。この場合、２つのソースオペランドｍｍ１とｍｍ２／ｍ６４が付随する命令である。本実施形態の命令は、複数のより小さいデータ要素から構成される６４ビットＰａｃｋｅｄデータブロックにより実行される。この場合、各データ要素は１６ビットまたはワードの長さを有する。合計６４ビットを形成する４つのワードが各Ｐａｃｋｅｄデータブロックに含まれうる。第１ソースオペランド「ｍｍ１」は６４ビットMMXレジスタである。本実施例では、第１ソースオペランドからの６４ビットMMXレジスタ「ｍｍ１」はまた、Ｐａｃｋｅｄ乗算上位丸めシフト処理の結果の宛先である。本例における第２ソースオペランド「ｍｍ２／ｍ６４」は、６４ビットMMXレジスタ（ｍｍ２）あるいは６４ビットメモリ位置（ｍ６４）でありうる。
【００６４】
以下で説明される例は一般に６４ビット長オペランドとデータブロックに関するものであるが、乗算上位丸めシフト命令の実施例はまた１２８ビットＰａｃｋｅｄデータブロックにより処理されうる。例えば、一実施例の命令フォーマットは、ＰＭＵＬＨＲＳＷｘｍｍ１、ｘｍｍ２／ｍ１２８として表すことができる。この場合における２つのソースオペランドはそれぞれ１２８ビット長であり、それぞれは１６ビットワードサイズの８つのデータ要素から構成される。第１ソースオペランド「ｘｍｍ１」は１２８ビットXMMレジスタである。本実施例において、XMMレジスタ「ｘｍｍ１」はまた結果の宛先である。本実施例における第２ソースオペランド「ｘｍｍ２／ｍ１２８」は、１２８ビットXMMレジスタ（ｘｍｍ２）または１２８ビットメモリ一（ｍ１２８）である。本実施例では、各データブロックは符号付き整数を含むことができる。一実施例では、符号付き整数は２の補数フォーマットである。
【００６５】
さらに、ここで説明される実施例はワードサイズのデータ要素から構成されるＰａｃｋｅｄデータブロックに関するものであるが、他の様々なサイズのデータ要素もまた考慮される。例えば、Ｐａｃｋｅｄ乗算上位丸めシフト命令の他の実施例が、バイト、ダブルワードまたはクアドワードの長さを有するデータ要素に対し実行されてもよい。同じように、データオペランドの長さは６４及び１２８に制限されない。例えば、他の実施例による命令は２５６ビット長のＰａｃｋｅｄオペランドに対し実行されうる。
【００６６】
図４Ｂは、本発明によるデータオペランドに対するＳＩＭＤ整数乗算上位丸めシフト処理を実行するための論理の一実施例のブロック図である。本実施例による乗算上位丸めシフト処理（または簡単化のため、乗算上位）のためのＰＭＵＬＨＲＳＷは、第１データオペランドＤＡＴＡＡ４１０と第２データオペランドＤＡＴＡＢ４２０の２つの情報から開始される。一実施例では、ＰＭＵＬＨＲＳＷ乗算上位命令は１つのマイクロ処理に復号化される。他の実施例では、データオペランドに乗算上位処理を実行するため、当該命令は可変数のマイクロｏｐに復号化される。
【００６７】
ここで、ＤＡＴＡＡ４１０、ＤＡＴＡＢ４２０とＲＥＳＵＬＴＡＮＴ４４０は、以下に限定されるものではないが、一般にオペランドあるいはデータブロックと呼ばれ、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、ＤＡＴＡＡ４１０とＤＡＴＡＢ４２０は６４ビット幅のMMXレジスタ（または、いくつかの例では、「ｍｍ」と呼ばれる）である。特定の実施形態に応じて、データオペランドは１２８または２５６ビットのような他の幅とすることができる。第１オペランド４１０と第２オペランド４２０は、ｘ個のデータ要素を含むデータブロックであり、各データブロックが１バイト（８ビット）である場合、それぞれが合計８ｘビット幅を有する。従って、各データセグメントは８ｘビット幅となる。ここでｘが８であるとき、各オペランドは８バイトまたは６４ビッ

ト幅である。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等のデータ要素幅であってもよい。

【００６８】

本実施例における第１Ｐａｃｋｅｄオペランド４１０は、４つのデータ要素Ａ３、Ａ２、Ａ１及びＡ０から構成される。第２Ｐａｃｋｅｄオペランド４２０はまた、４つのデータ要素Ｂ３、Ｂ２、Ｂ１及びＢ０から構成される。ここでのデータ要素は同じ長さを有し、それぞれ１ワード（１６ビット）の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、各データセグメントが１バイト（８ビット）からなるより長い１２８ビットオペランドにおいて処理され、１２８ビット幅オペランドは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。本発明の実施例は特定の長さのデータオペランドあるいはデータセグメントに制限されず、各実施形態に適したサイズとすることができる。

【００６９】

オペランド４１０と４２０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の組み合わせたものに配置される。データオペランド４１０と４２０は乗算上位丸めシフト命令と共に、プロセッサの実行ユニットの乗算上位丸めシフト計算論理４３０に送られる。ＰＭＵＬＨＲＳＷ命令が実行ユニットに達するまで、プロセッサパイプラインにおいて当該命令は前もって復号されるべきである。従って、乗算上位命令はマイクロ処理（ｕｏｐ）あるいは他の復号化フォーマットの形式に従いうる。本実施例では、２つのデータオペランド４１０と４２０は、乗算上位丸めシフト計算論理４３０において受信される。本実施例は６４ビット幅オペランドに対し実行されるので、仮のスペース４３１は１２８ビット幅の中間結果の積を保持する必要がある。１２８ビット幅のデータオペランドに対し、２５６ビット幅の仮スペースが必要とされる。

【００７０】

本実施例の論理４３０はまず、積Ａ×Ｂを得るために各要素位置において対応するデータ値を掛け合わせる。４つの位置に対するＡ×Ｂの各中間３２ビット値はそれぞれ１８ビットに切り捨てられる。本実施例では、切り捨て（ｔｒｕｎｃａｔｉｏｎ）は各３２ビット値を１４ビットだけ右シフトし、これらのビットを取り除くことにより行われる。これおにより、各仮の値には１８ビットが残される。１つの「１」が丸め処理のため本実施例の最下位ビットに付け加えられる。丸められた各値の最上位ビットのすぐ右の１６ビットが、結果４４０の各データ要素位置に出力される。本実施例における最左データ要素位置において、当該結果は「（（Ａ３×Ｂ３）＞＞１４）＋１」のビット［１６：１］に等しい。丸められた結果のビット［１６：１］の選択は、小数演算と同じようにこの値を適切にスケーリングする。

【００７１】

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対し実行可能である。従って、本発明の実施例は、特定の長さのオペランド、データセグメント及びシフトカウントに制限されるものでなく、各実施形態に適したサイズとすることができる。

【００７２】

実行時、一実施例のＰａｃｋｅｄ整数乗算上位丸めシフト命令は、第１ソースオペランド及び第２ソースオペランドのＰａｃｋｅｄ符号付き整数ワードのＳＩＭＤ符号付き１６ビット×１６ビットの乗算を実行し、正確な３２ビット中間積を生成する。一実施例における中間積はまず上位１８ビットに切り捨てられる。この１８ビットの選択により、１８ビットの中間精度が与えられる。１８ビット値の最下位ビットに「１」を付け加えること

により、この切り捨てられた値に対し丸め処理が行われる。言い換えると、丸め処理は、もとの３２ビット中間積の第１４ビットにおけるビット値に「１」を加えるというものである。１８ビットの値の最上位ビットのすぐ右に１６ビットを選択することにより最終的な結果が得られる。本実施例では、結果の各値は１つの符号ビットを含んでいる。本実施例の結果の各データ要素は「１．１５」の固定点整数フォーマットを有することが可能である。本実施例の乗算上位丸めシフト命令は、各丸めシフトされた中間３２ビット値の１６ビットを宛先オペランドの適当な位置に格納する。

【００７３】

本実施例において、これと他のデータ要素位置の結果が、ソースデータオペランドと同じサイズのデータブロック結果にＰａｃｋされる。例えば、ソースＰａｃｋｅｄデータオペランドが６４または１２８ビット幅である場合、結果として得られるＰａｃｋｅｄデータブロックもまたそれぞれ６４または１２８ビット幅となる。さらに、符号処理に対するソースデータオペランドはレジスタまたはメモリ領域から得られる。本実施例では、結果として生じるＰａｃｋｅｄデータブロックは、ソースデータオペランドの１つのためにＳＩＭＤレジスタのデータを上書きする。

【００７４】

図４Ｂは、選択されたデータ要素位置に対する整数乗数上位丸めシフト処理の動作のブロック図である。ＤＡＴＡ　ＥＬＥＭＥＮＴ　Ａ４５０は第１ソースオペランドからのものである。ＤＡＴＡ　ＥＬＥＭＥＮＴ　Ｂ４５２は第２ソースオペランドからのものである。本実施例の乗算上位丸めシフト処理４５４は、中間値ＴＥＭＰ４５６の積を生成するため、データ要素を掛け合わせることにより開始される。２つの１６ビット幅ソースデータ要素に対し、積は３２ビット中間値である。本実施例において、ＴＥＭＰ４５６の最上位１８ビットが丸めスケーリング処理に利用される。１８ビットを維持することにより、計算におけるさらなる精度が達成できる。乗算上位丸めシフト処理４５４は、最新の中間値４５８を得るために中間値４５６に対し丸め及びスケーリング処理実行することにより継続される。本実施例では、３２ビット中間値ＴＥＭＰ４５６の第１４ビットに「１」を加えることにより丸め処理は行われる。ところで、３２ビット値の第１４ビットはまた、興味のある１８ビット幅部分の最下位ビットでもある。中間値をスケーリングするために、３２ビットの丸められた値に対しシフト処理が行われる。最新の中間値４５８に到達するため、１ビットの左シフトが丸められた値に対し実行される。最新の中間値４５８は切り捨てられ、ＲＥＳＵＬＴ４６０が得られる。本実施例において、興味のあるビットは最新の３２ビット中間値４５８の上位１６ビットであり、ＲＥＳＵＬＴ４６０として格納される。下位１６ビットは切り捨て処理において切り捨てられる。

【００７５】

図５は、本発明による乗算上位丸めシフト処理を実行する回路５００の一実施例のブロック図である。本実施例の回路５００は、ベクトル複合整数ユニット（ｖｅｃｔｏｒ　ｃｏｍｐｌｅｘ　ｉｎｔｅｇｅｒ　ｕｎｉｔ）内に設けられる。この整数ユニットはＰＭＵＬＨＲＳＷ命令を１２８ビットオペランドによる実施形態のため、それぞれが１６ビット×１６ビットの乗算を実行する８つのパートに分割される。６４ビットオペランドによる実施形態では、４つのパートが必要とされる。図５では、ＳＲＣ　Ｙ　ＥＬＥＭＥＮＴ５０２が基数４のブースリコード（ｒａｄｉｘ－４　ｂｏｏｔｈ　ｒｅｃｏｄｅ）ブロック５０４に送られる。ＳＲＣ　Ｘ　ＥＬＥＭＥＮＴ５０２はブースマックス（ｂｏｏｔｈ　ｍｕｘ）５０８において受け取られる。ブースマックスは、９つの部分積ベクトル５０９を生成する。

【００７６】

手計算による掛け算処理において、あるオペランド（Ａ）の最下位ビットを抽出し、このビットをもう一方のオペランド（Ｂ）の各桁とビット単位で掛け合わせることにより処理が開始される。乗算対象Ａの各ビットに対し、１行の結果が生成される。これらの行のそれぞれは部分積として知られている。例えば、

【表１】

```
      1001 (9)
    x 0110 (6)
---------
      0000
     1001
    1001
+ 0000
---------
  0110110 (54)
```

大きな数の乗算ではすべての部分積を処理するために、多くのハードウェアが必要とされるので、計算の簡単化のために一実施例においてブースリコード技術が実行される。ブースリコード処理では、部分積の半数をわずかに上回る（Ｎビット／２＋１）が、手計算と同じように生成される。例えば、上記４つの部分積を得る代わりに、ブースリコード処理は３つの部分積を生成する。従って、１６×１６の乗数に対して、加えられるべき部分積は「１６／２＋１」、すなわち９となる。この方法は基数４とここで呼ばれる。各１６ビット乗算配列は、基数４のブース符号化配列である。ブース符号化処理では９つの部分積が生成され、これは桁上げ和加算器（ＣＳＡ）ツリー構造と加算器により低減された。一実施例では、ＣＳＡツリーの１６ビット配列構造の全体は、以下のようなものである。

【００７７】

【表２】

```
4      4         33          2    1    10          0 bit
76543210987654321098765432109876543210987654321098765432109876543210
                                pp                                  rowI_p1
                  ls=----------------pp00000000000000   I 8
                    ls=--------------pp000000000000   H 7
                      ls=------------pp0000000000   G 6
                        ls=----------pp00000000   F 5
                          ls=--------pp000000   E 4
                            ls=------pp0000   D 3
                              ls=----pp00   C 2
                                ls=--pp   B 1
                                  sSS=-   A 0

                             |-------------|
```

本実施例は負の乗算を扱えるよう構成されている。「Ｓ」は符号を表し、「Ｐ」は前の部分積の下位２ビットを記述するのに利用されている。例えば、部分積１の「ｐｐ」は部分積０の最下位２ビットである。先頭の符号拡張の本質は、符号ビットをロールオフすることである。これは、乗算前に、負の数を正にする２の補数のビット反転と同様である。同じように、「Ｐ」ビットの本質は、負から正への変換の２の補数反転に対し＋１を与えるということである。

【００７８】

ビット［３１：１６］は、乗算の上位結果ビットとしてみなすことができる。しかしながら、乗算上位丸めシフトでは、最終結果前に丸め処理とシフト処理が扱われる。一実施例では、丸め処理は配列のビット位置１４に「１」を加えることに関する。しかしながら、部分積ツリーの第１４ビットには「１」を容易に加えるための空き位置がない。第８行では、ビット１３、１２及び１１が空き位置である。同じように、第７行のビット１１に空き位置がある。以下のＲビットに示されるように、「１」をこれら４つの位置のすべてに加えることは「１」をビット位置１４まで拡げることになる。本実施例の丸め技術では

、ＣＳＡ圧縮ツリー５１０は、以下のようになる。

【００７９】

【表３】

```
4       4         33          2    1    10          0
765432109876543210987654321098765432109876543210
                                 pp                    rowI_pl
                  ls=----------------ppRRR00000000000   I 8
                    ls=----------------ppR00000000000   H 7
                      ls=----------------pp0000000000   G 6
                        ls=----------------pp00000000   F 5
                          ls=----------------pp000000   E 4
                            ls=----------------pp0000   D 3
                              ls=----------------pp00   C 2
                                ls=----------------pp   B 1
                                  sSS=----------------   A 0

                        |---pmulhrsw---|
```

本発明の実施例はＣＳＡを利用して、３２ビット加算器５１４前に部分積の項数を９から２に減らすのに役立つ。一実施例では、ＣＳＡ圧縮ツリーは部分積の項数を（４：２ＣＳＡを利用して）まず９から６に、その後６から４に、最後に４から２に減らしていく。このテクニックは９つの３２ビット加算器に対する必要性を回避する。本実施例におけるＣＳＡツリー５１０の出力は、２に減じられた部分積の項である。１つは最後のＣＳＡの合計項であり、もう１つは桁上がり「ｃａｒｒｙ　ｏｕｔ」の項である。完全な結果を得るためこれら２つの項を論理的に加えるために、桁上がり項は合計項と適切に一致するよう１ビット左にシフトされねばならない。例えば、桁上がり項の最下位ビットであるビット０は、合計項のビット１と並べられる必要がある。

【００８０】

３２ビット加算器５１４は、ＳＵＭ５１２とＣＡＲＲＹ５１１を加算することにより、ＦＵＬＬ　ＲＥＳＵＬＴ５１５を生成する。本実施例のＳＵＭ５１２はＳＵＭ［３１：０］である。Ｃａｒｒｙ５１１は１ビット左にシフトされたＣａｒｒｙ［３０：０］である。本実施例に関連するビットは、ビット［３０：１５］である。これら１６ビットが、上記乗算の積から１ビットだけシフトされる。回路５００の本実施例において、このシフト処理は、結果マックス５１８と結果マックスでコード５１６により実現される。従って、符号付き整数乗算上位丸めシフト処理のＲＥＳＵＬＴＡＮＴ５２０は、ＦＵＬＬ　ＲＥＳＵＬＴ５１５の最上位ビットのすぐ右の１６ビット、すなわち、ＦＵＬＬ　ＲＥＳＵＬＴ［３０：１６］となる。本実施例において、データ要素の各ペアに対して８つの配列構造のそれぞれからの結果が、最終的な１２８ビットの結果を得るため連結される。

【００８１】

図６Ａは、本発明の第１実施例によるＰａｃｋｅｄ乗算上位丸めシフト命令の動作を示す。６４ビット幅のソースオペランドＤＡＴＡ　Ａ６０１は、それぞれが１６進数４７９Ｃ$_{16}$、１ＡＦ７$_{16}$、Ｃ０００$_{16}$及び０２００$_{16}$を格納する４つのデータ要素６０２、６０３、６０４及び６０５から構成される。同様に、６４ビット幅のソースオペランドＤＡＴＡ　Ｂ６１１は、それぞれが１６進数Ｄ７６Ｅ$_{16}$、２ＢＣ５$_{16}$、Ｃ０ＦＦ$_{16}$及び０２２０$_{16}$を有する４つのデータ要素６１２、６１３、」６１４及び６１５から構成される。ソースオペランドとしてＤＡＴＡ　Ａ６０１とＤＡＴＡ　Ｂ６１１と共に、本発明の一実施例によるＰａｃｋｅｄ乗算上位丸めスケーリング命令は、ＲＥＳＵＬＴＡＮＴオペランド６２１を生成する。本実施例のＰａｃｋｅｄ乗算上位丸めスケーリング処理６２０は、ソースデータ要素の対応する各ペアに対し結果を生成する。本実施例では、ＲＥＳＵＬＴＡＮＴ６２１の４つのデータ要素は、１６進数Ｅ９４Ｅ$_{16}$６２２、０９３８$_{16}$６２３、１Ｆ８１$_{16}$６２４及び０００９$_{16}$６２５を有する。

【００８２】

図６Ｂは、図６Ａの特定のデータ要素位置におけるＰａｃｋｅｄ乗算上位命令のさらな

る詳細な動作を示す。図６Ａの例から継続して、ここでは左から２番目のデータ要素位置がより詳細に説明される。ＤＡＴＡ　Ａ６０１の第２最左データ要素６０３の値は１ＡＦ７$_{16}$（あるいは、２進数では００１　１０１０　１１１１　０１１１）である。ＤＡＴＡ　Ｂ６１１の第２最左データ要素６１３の値は２ＢＣ５$_{16}$（あるいは、２進数では００１０　１０１１　１１００　０１０１）である。Ｐａｃｋｅｄ乗算上位丸めスケーリング処理において、これら２つの値はまず掛け合わされ、０４９Ｃ３Ｄ１３$_{16}$（０００００１００　１００１　１１００　００１１　１１０１　０００１　００１１$_{2}$）の積６３１が得られる。この積６３１は、第１の仮の中間値ＴＥＭＰ６３０として扱われる。

【００８３】

この処理の丸め部分６３３が積６３１に対し実行される。本実施例では、丸め処理は、「１」を積６３１の第１４ビット６３２に加えることである。丸め処理６３３の結果６３４は、新たなＴＥＭＰ６３０を生成する。丸め処理による結果６３４は０４９Ｃ７Ｄ１３$_{16}$（０００００１００　１００１　１１００　０１１１　１１０１　０００１　００１１$_{2}$）を有する。丸め処理の結果６３４はスケーリングされ、本実施例における所望の結果が得られる。ここでのスケーリング処理６３６は、ＴＥＭＰ６３０の丸め処理の結果６３４の１ビットの左シフトとして実行される。従って、ビット３０から１５がビット位置３１から１６にシフトアップされる。ＴＥＭＰ６３０は１６ビット値に切り捨てられ、丸めシフトされた値の最上位１６ビット（上位部分）がＲＥＳＵＬＴＡＮＴ６２３として出力される。ＲＥＳＵＬＴＡＮＴ６２３は、ＰａｃｋｅｄＲＥＳＵＬＴＡＮＴ６１２の左から２番目のデータ要素位置である。本実施例では、ＲＥＳＵＬＴＡＮＴ６２３は０９３８$_{16}$（０００００１００１　００１１　１０００$_{2}$）である。

【００８４】

６４ビットオペランドのペアの第２データ要素位置でのＰａｃｋｅｄ乗算上位丸めスケーリング（ＰＭＵＬＨＲＳＷ）処理の例が以下のように示される。

【００８５】

【表４】

```
0x1AF7 x 0x2BC5 = 0x0938

temp = 0x1AF7 x 0x2BC5 = 0x049C3D13
                       = 0000 0100 1001 1100 0011 1101 0001 0011

     ROUND             + 0000 0000 0000 0000 0100 0000 0000 0000
                         -----------------------------------

temp (rounded result)  = 0000 0100 1001 1100 0111 1101 0001 0011

     SHIFT << 1 BIT

temp (shifted result)  = 0000 1001 0011 1000 1111 1010 0010 0110

     TRUNCATE            ---- ---- ---- ----

               RESULT = 0000 1001 0011 1000 = 0x0938
```

上記例では、ソースデータオペランドの一方または両方は、ＭＭＸ／ＳＳＥ技術により可能なプロセッサにおける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とすることができる。実施形態に応じて、これらのレジスタは６４／１２８／２５６ビット幅とすることができる。同様に、ソースオペランドの一方あるいは両方は、レジスタ以外のメモリ領域とすることができる。一実施例において、結果の宛先はＭＭＸあるいはＸＭＭデータレジスタである。さらに、結果の宛先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、一アーキテクチャでは、乗算上位丸めシフト命令は、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宛先は、この場合、第１ソースオペランドＭＭ１のレジスタとするこ

とができる。

【００８６】

図７Ａは、積の上位部分を得るためのＰａｃｋｅｄデータオペランドに対する整数乗算丸めシフト処理を実行する方法の一実施例を示すフローチャート７００である。長さＬを使って、オペランドとデータブロックの幅が表される。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、バイト数ワード数などに関する長さを示すのに利用される。ブロック７１０において、長さＬの第１データオペランドＡは、Ｐａｃｋｅｄ整数乗算上位丸めシフト処理の実行のため受信される。ブロック７２０において、ＰＭＵＬＨＲＳＷ処理のための長さＬの第２データオペランドＢが受信される。ブロック７３０において、乗算上位丸めシフトの実行命令が処理される。

【００８７】

本実施例のブロック７３０における乗算上位丸めシフト処理の詳細が、各データ要素位置に対して発生するものに関してさらなる説明が与えられる。一実施例では、結果として生じるＰａｃｋｅｄデータ要素位置のすべてに対する乗算上位丸めシフト処理は並列に処理される。他の実施例では、データ要素のある部分が同時に処理される。ブロック７３１におおいて、オペランドＡからの要素の値とオペランドＢからの要素の値を掛け合わせることにより仮の値ＴＥＭＰが計算される。ブロック７３２において、この仮の値は丸められる。一実施例では、仮の値の上位１８ビットがより高い精度のための計算に利用される。他の実施例では、他の個数のビットが対象となるかもしれない。ブロック７３２の丸め処理の後、ブロック７３３において仮の値がスケーリングされる。本実施例では、スケーリング処理は仮の値を１ビットだけ左にシフトすることである。ブロック７３４において、仮の値は必要なビット数に切り捨てられ、結果の値として宛先に格納される。ソースデータ要素の異なるペアのそれぞれに対する結果の値は、結果として生じるＰａｃｋｅｄオペランドのソース要素ペアに対応する適切なデータ要素位置に配置される。

【００８８】

図７Ｂは、Ｐａｃｋｅｄ整数乗算丸めシフト処理の結果として得られる積の関連する上位部分を獲得する方法の他の実施例を示すフローチャートである。本実施例において、オペランドはワードサイズのデータ要素から構成される。しかしながら、他の実施例は、例えば、バイト、ダブルワードあるいはクアドワードのような他のサイズのデータ要素により実現されるかもしれない。ブロック７４２において、乗算上位丸めシフト処理の制御信号が復号される。ブロック７４４において、当該処理におけるオペランドサイズの決定がチェックされる。一実施例では、オペランドサイズはブロック７４２で復号化された制御信号により決定することができる。例えば、オペランドサイズは命令により符号化することができる。オペランドサイズが６４ビット長であると判断されれば、ブロック７４６においてレジスタファイル及び／またはメモリがアクセスされ、データのある場所に応じてオペランドデータが取得される。一実施例では、ソースオペランドはＳＩＭＤレジスタ及び／またはメモリ領域にあるかもしれない。本実施例の６４ビット長のオペランドでは、各オペランドは４ワードサイズのデータ要素を有する。

【００８９】

ソースデータ要素のこれら４つのペアの計算が、ブロック７４７の４つの式のセットとして示される。第１式の「ＴＥＭＰ［３１：０］＝Ａ［１５：０］×Ｂ［１５：０］」は、ソースデータ要素の乗算を表す。第２式の「ＩＮＴ（ＴＥＭＰ［３１：０］＞＞１４）＋１」は、中間結果の丸め処理を表す。本実施例では、仮の値は１４ビット右シフトされ、最下位ビットに「１」が加えられる。言い換えると、中間結果の上位１８ビットが保持され、もとの第１４ビットに「１」が加えられる。第３式の「ＤＥＳＴ［１５：０］＝ＴＥＭＰ［１６：１］」は、丸められた結果のシフトおよび切り捨て処理を表す。この場合、結果として得られる各デーや要素はワードであり、１６ビットが必要とされる。残りの１８ビットのビット［１６：１］がここでは抽出される。本実施例では、左へのシフトは、最下位ビットのすぐ左の１６ビットをとることにより行われた。切り捨てられた値は当該データ要素位置の結果として格納される。ブロック７４７において、ビット範囲が当該

位置の正しい値により満たされる場合を除き（すなわち、［１５：０］、［３１：１５］、［４７：３２］及び［６３：４８］）、この３つの式が各データ要素位置に対し繰り返される。

【００９０】

ブロック７４４においてオペランドサイズが１２８ビット長であると判断されると、ブロック７４８においてレジスタファイル及び／またはメモリがアクセスされ、必要なオペランドデータが取得される。本実施例の１２８ビット長のオペランドに対し、各オペランドは８ワードサイズのデータ要素を有する。６４ビットパスと同様に、ブロック７４９においてソースデータ要素の８つのペアのそれぞれが上記３つの式のセットにより処理される。この１２８ビットパスの８つの式のセットに対する正しいビット範囲は、［１５：０］、［３１：１５］、［４７：３２］、［６３：４８］、［７９：６４］、［９５：８０］、［１１１：９６］及び［１２７：１１２］である。本実施例で説明される２つのパスは６４ビットオペランドと１２８ビットオペランドに関するものであるが、他の様々な長さのオペランドが他の実施例において利用されうる６つの１６ビット値は、それぞれが各データ要素位置に対応し、ＤＥＳＴの各自のデータ要素位置に格納される。

【００９１】

ＳＩＭＤ整数乗算丸めシフトのテクニックが開示された。特定の実施例が添付された図面と共に説明及び示されたが、このような実施例は単なる例示のものであり、発明の範囲を限定するものではない。また、本発明は例示及び説明された特定の構成及び配置に制限されるものではなく、当業者により本開示に基づき他の様々な変更を行うことができるであろう。進歩のスピードが速く、さらなる進展が容易に予想できないこのような技術分野では、開示された実施例は、本開示の原理あるいは添付されたクレームの範囲を逸脱することなく、技術の進歩により促進されるような修正が可能である。

【００９２】

本発明は上記特定の実施例に限定されるものではなく、本発明の要旨内において様々な変形・変更が可能である。

【図面の簡単な説明】

【００９３】

【図１Ａ】図１Ａは、本発明の一実施例による整数乗算上位丸めシフト処理のためのＳＩＭＤ命令を実行する実行ユニットを有するプロセッサから構成されるコンピュータシステムのブロック図である。

【図１Ｂ】図１Ｂは、本発明の他の実施例による他の一例となるコンピュータシステムのブロック図である。

【図１Ｃ】図１Ｃは、本発明の他の実施例によるさらなる他の一例となるコンピュータシステムのブロック図である。

【図２】図２は、本発明によるＰａｃｋｅｄ整数乗算上位丸めシフト処理を実行する論理回路を有する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。

【図３Ａ】図３Ａは、本発明の一実施例によるマルチメディアレジスタの様々なＰａｃｋｅｄデータタイプ表現を示す。

【図３Ｂ】図３Ｂは、他の実施例によるＰａｃｋｅｄデータタイプを示す。

【図３Ｃ】図３Ｃは、Ｐａｃｋｅｄ乗算上位丸めシフト命令の処理符号化（オペコード）フォーマットの一実施例を示す。

【図３Ｄ】図３Ｄは、他の処理符号化フォーマットを示す。

【図３Ｅ】図３Ｅは、さらなる他の処理符号化フォーマットを示す。

【図４Ａ】図４Ａは、本発明によるデータオペランドに対するＳＩＭＤ整数乗算上位丸めシフト処理を実行する論理の一実施例のブロック図である。

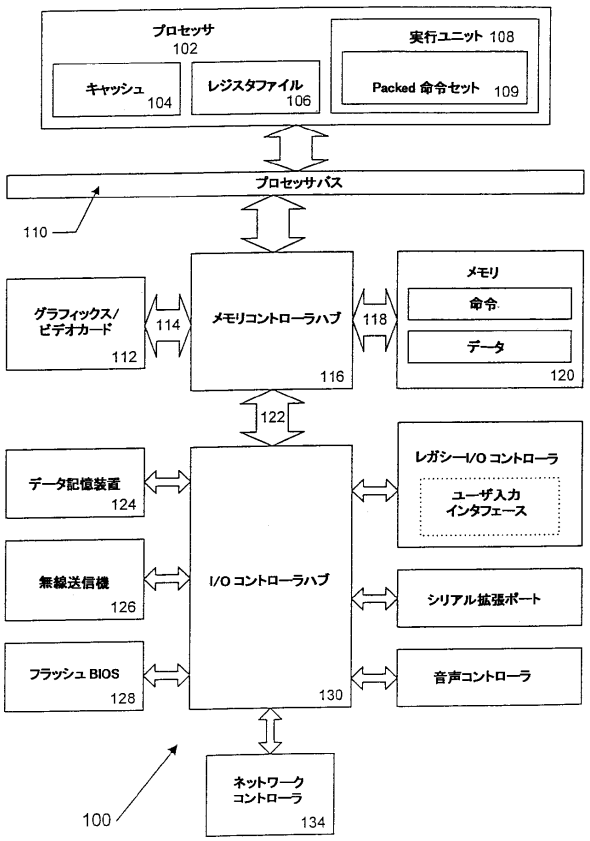
【図４Ｂ】図４Ｂは、選択されたデータ要素位置に対する整数乗算上位丸めシフト処理の動作のブロック図である。

【図５】図５は、本発明による乗算上位丸めシフト処理を実行する回路の一実施例のブロック図である。

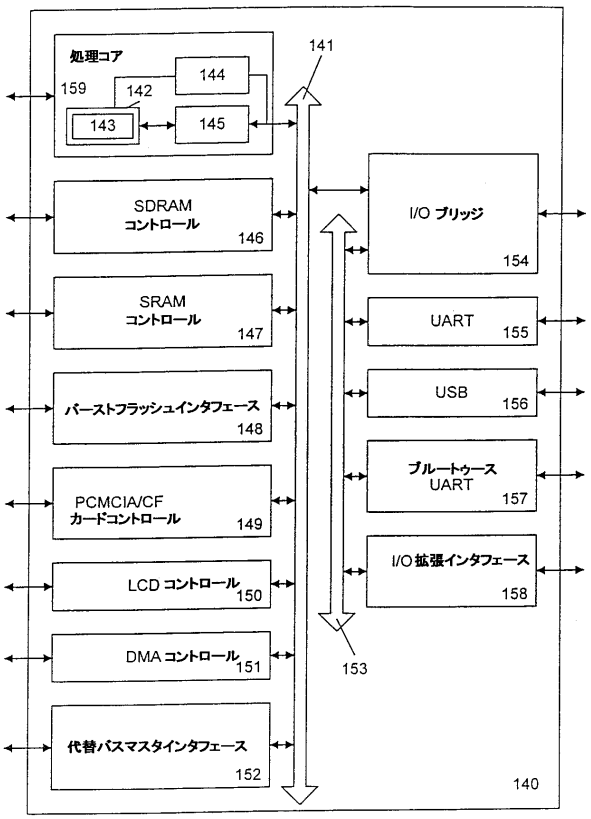
【図６Ａ】図６Ａは、本発明の第１実施例によるＰａｃｋｅｄ乗算上位丸めシフト命令の動作を示す。
【図６Ｂ】図６Ｂは、図６Ａの特定のデータ要素位置におけるＰａｃｋｅｄ乗算上位命令のさらなる詳細な動作を示す。
【図７Ａ】図７Ａは、積の上位部分を取得するためのＰａｃｋｅｄデータオペランドに対する整数乗算丸めシフト処理を実行する方法の一実施例を示すフローチャートである。
【図７Ｂ】図７Ｂは、Ｐａｃｋｅｄ整数乗算丸めシフト処理の結果として生じる積の関連する上位部分を取得する方法の他の実施例を示すフローチャートである。
【符号の説明】
【００９４】
１００、１４０、１６０　コンピュータシステム
１０２、１６６、２００　プロセッサ
１０４、１６７　キャッシュ
１０６、２０８、２１０　レジスタファイル
１０８　実行ユニット
１０９　Ｐａｃｋｅｄ命令セット
１１０　プロセッサバス
１１２　グラフィックス／ビデオカード
１１４　ＡＧＰインターコネクト
１１６　メモリコントローラハブ（ＭＣＨ）
１１８　メモリインタフェース
１２０　メモリ
１２２　専用ハブインタフェースバス
１２４　データ記憶装置
１２６　無線送信機
１２８　フラッシュＢＩＯＳ
１３０　Ｉ／Ｏコントローラハブ（ＩＣＨ）
１３４　ネットワークコントローラ
１４１　バス
１４２、１６２　実行ユニット
１４３　Ｐａｃｋｅｄ命令セット
１４４、１６５　デコーダ
１４５、１６４　レジスタファイル
１４６　ＳＤＲＡＭコントロール
１４７　ＳＲＡＭコントロール
１４８　バーストフラッシュメモリインタフェース
１４９　ＰＣＭＣＩＡ／ＣＦカードコントロール
１５０　ＬＣＤコントロール
１５１　ＤＭＡコントロール
１５２　代替バスマスタインタフェース
１５３　Ｉ／Ｏバス
１５４　Ｉ／Ｏブリッジ
１５５　ＵＡＲＴ
１５６　ＵＳＢ
１５７　ブルートゥースＵＡＲＴ
１５８　Ｉ／Ｏ拡張インタフェース
１５９、１７０　処理コア
１６１　ＳＩＭＤコプロセッサ
１６３　命令セット
１６８　Ｉ／Ｏシステム

１６９　無線インタフェース
２０１　フロントエンド
２０２　高速スケジューラ
２０３　アウト・オブ・オーダーエンジン
２０４　低速／通常浮動小数点スケジューラ
２０６　シンプル浮動小数点スケジューラ
２１１　実行ブロック
２１２、２１４　アドレス生成ユニット（ＡＧＵ）
２１６、２１８　高速ＡＬＵ
２２０　低速ＡＬＵ
２２２　浮動小数点ＡＬＵ
２２４　浮動小数点移動ユニット
２２６　命令プリフェッチャ
２２８　命令デコーダ
２３０　トレースキャッシュ
２３２　マイクロコードＲＯＭ
２３４　ｕｏｐキュー
４３０　乗算上位丸めシフト計算論理

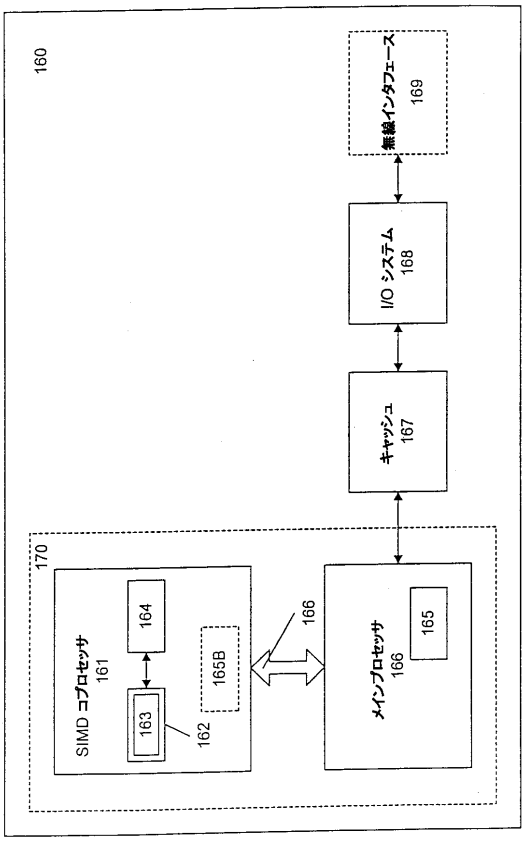
【図１Ａ】



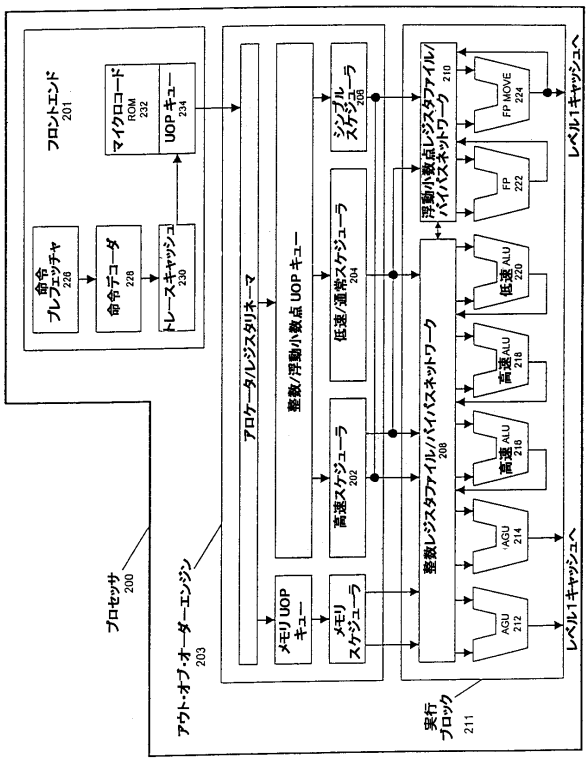
【図１Ｂ】



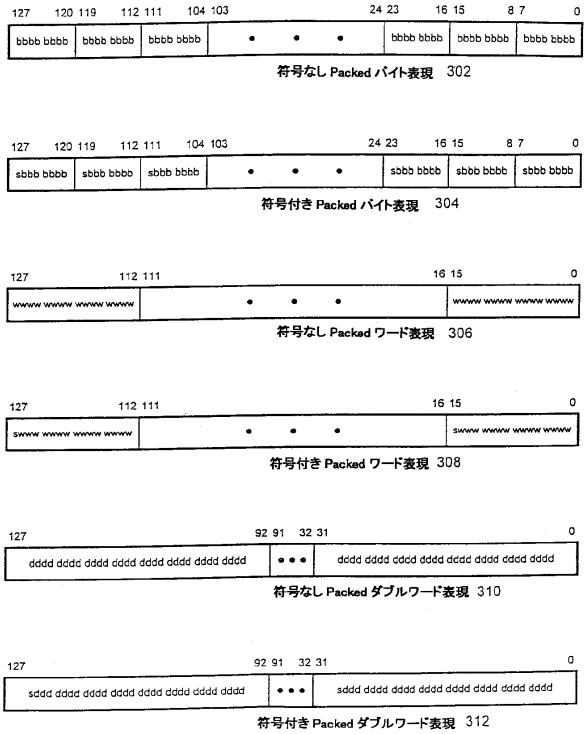
【図１Ｃ】



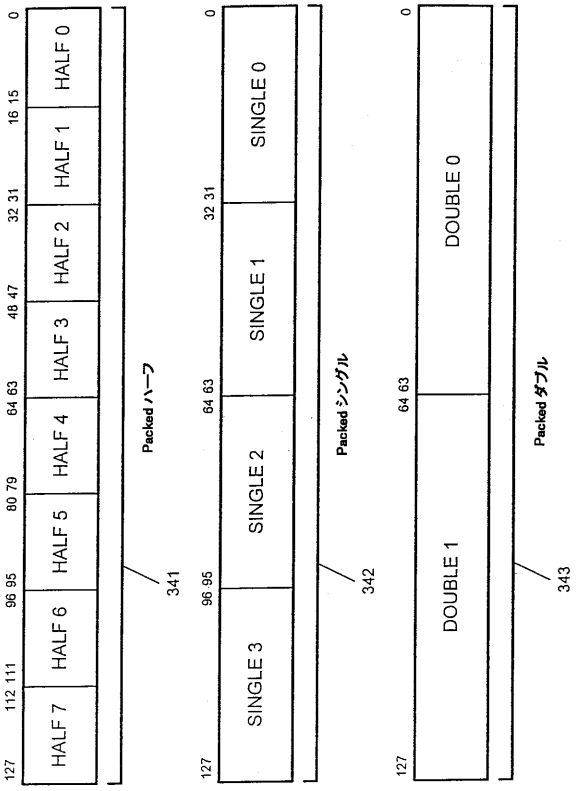
【図２】



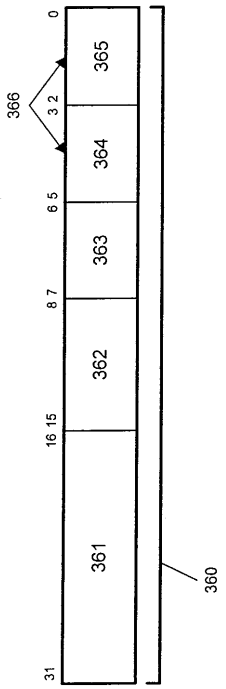
【図３Ａ】



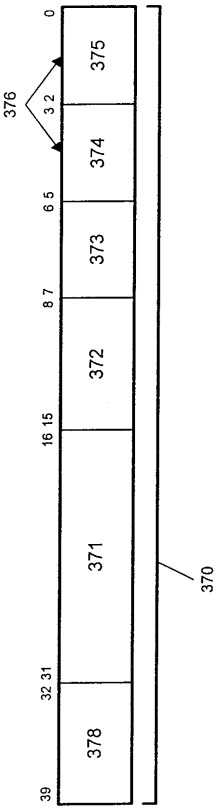
【図３Ｂ】



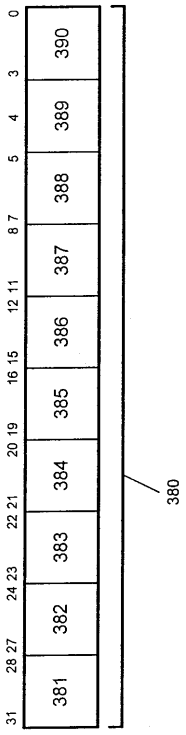
【図３Ｃ】



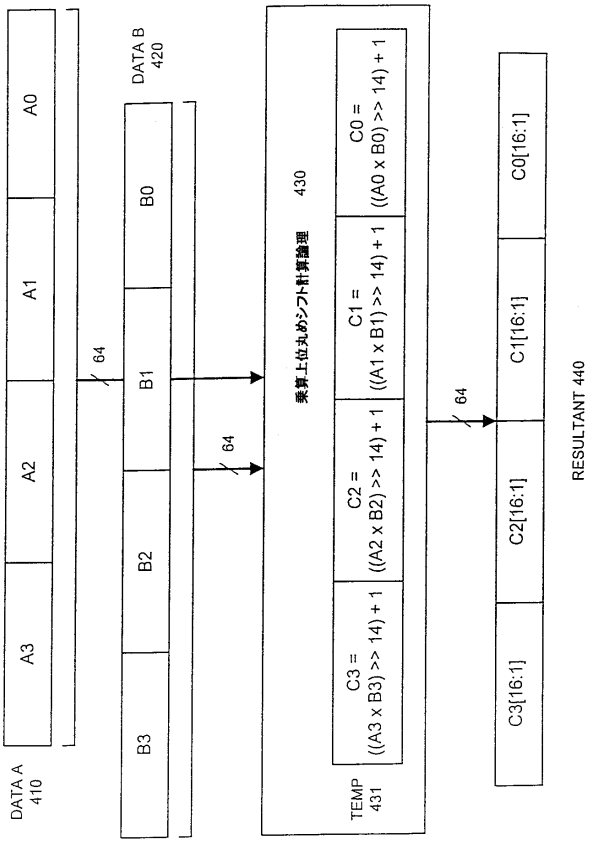
【図３Ｄ】



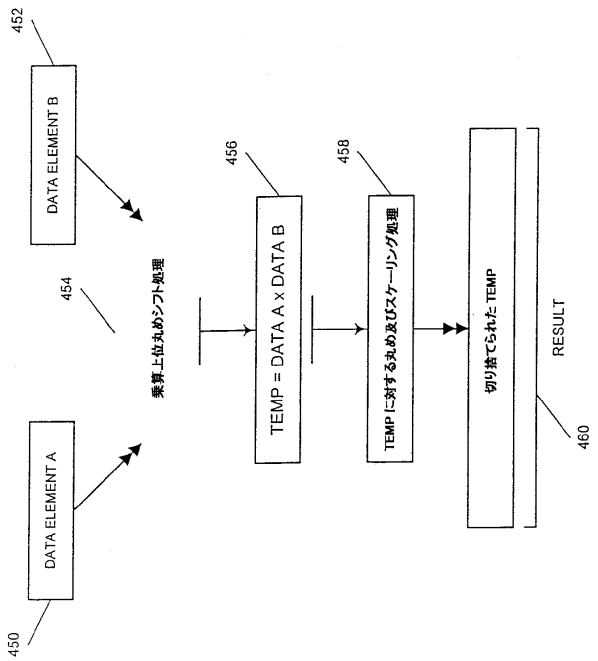
【図３Ｅ】



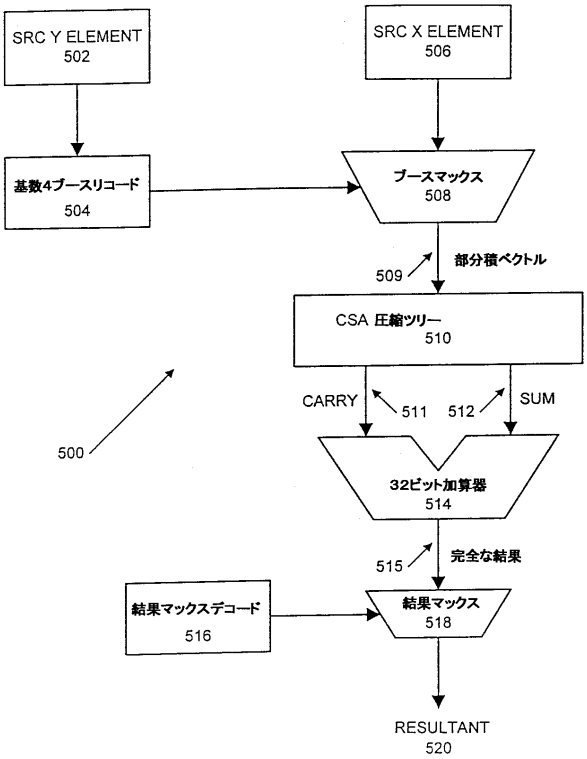
【図４Ａ】



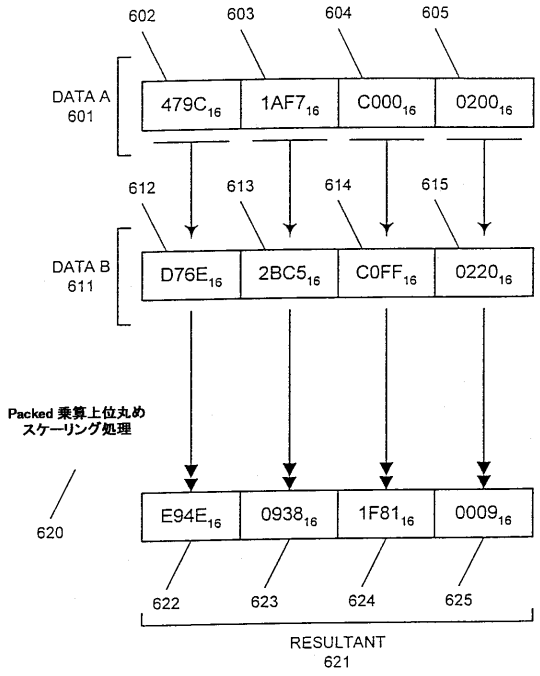
【図４Ｂ】



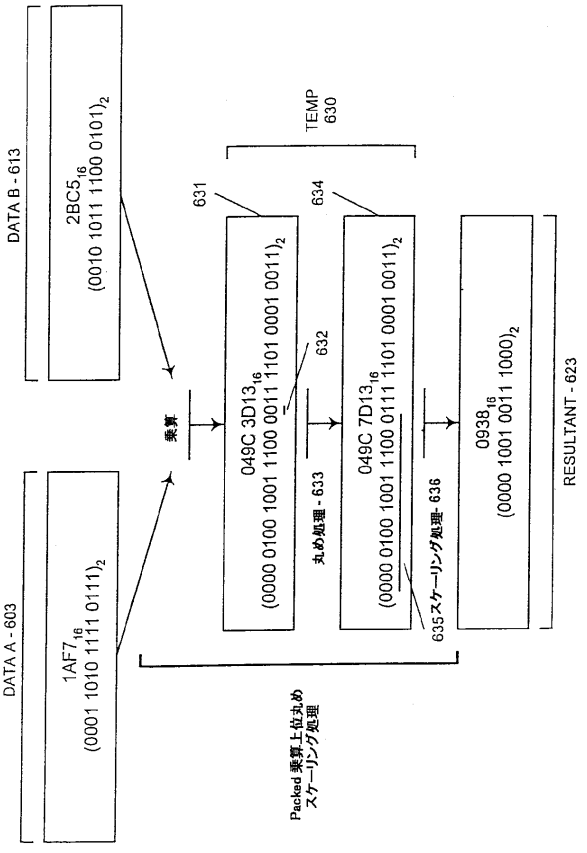
【図５】



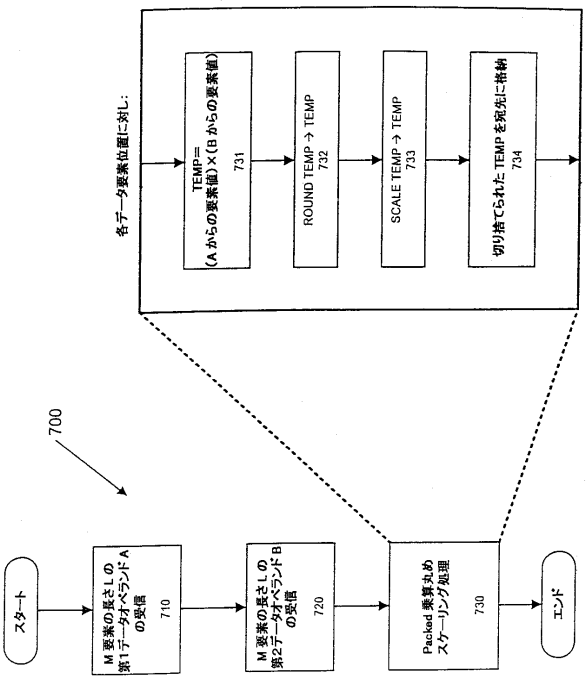
【図６Ａ】



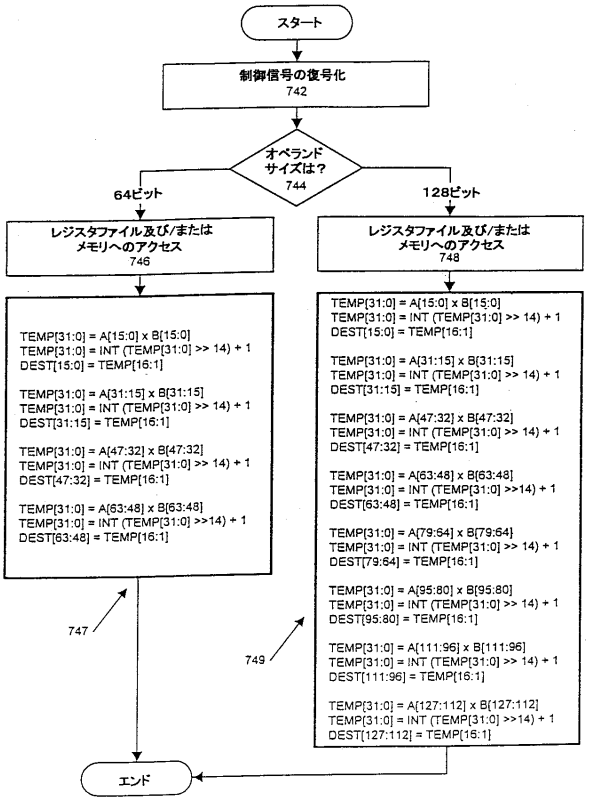
【図６Ｂ】



【図７Ａ】


各データ要素位置に対し：
TEMP=
(Aからの要素値)×(Bからの要素値)
731
ROUND TEMP → TEMP
732
SCALE TEMP → TEMP
733
切り捨てられたTEMPを宛先に格納
734
700
スタート
M要素の長さLの
第1データオペランドA
の受信
710
M要素の長さLの
第2データオペランドB
の受信
720
Packed乗算丸め
スケーリング処理
730
エンド

【図７Ｂ】


スタート
制御信号の復号化
742
オペランド
サイズは？
744
64ビット
128ビット
レジスタファイル及び/または
メモリへのアクセス
746
レジスタファイル及び/または
メモリへのアクセス
748
TEMP[31:0] = A[15:0] x B[15:0]
TEMP[31:0] = INT (TEMP[31:0] >> 14) + 1
DEST[15:0] = TEMP[16:1]
TEMP[31:0] = A[31:15] x B[31:15]
TEMP[31:0] = INT (TEMP[31:0] >> 14) + 1
DEST[31:15] = TEMP[16:1]
TEMP[31:0] = A[47:32] x B[47:32]
TEMP[31:0] = INT (TEMP[31:0] >> 14) + 1
DEST[47:32] = TEMP[16:1]
TEMP[31:0] = A[63:48] x B[63:48]
TEMP[31:0] = INT (TEMP[31:0] >>14) + 1
DEST[63:48] = TEMP[16:1]
TEMP[31:0] = A[15:0] x B[15:0]
TEMP[31:0] = INT (TEMP[31:0] >> 14) + 1
DEST[15:0] = TEMP[16:1]
TEMP[31:0] = A[31:15] x B[31:15]
TEMP[31:0] = INT (TEMP[31:0] >> 14) + 1
DEST[31:15] = TEMP[16:1]
TEMP[31:0] = A[47:32] x B[47:32]
TEMP[31:0] = INT (TEMP[31:0] >> 14) + 1
DEST[47:32] = TEMP[16:1]
TEMP[31:0] = A[63:48] x B[63:48]
TEMP[31:0] = INT (TEMP[31:0] >>14) + 1
DEST[63:48] = TEMP[16:1]
TEMP[31:0] = A[79:64] x B[79:64]
TEMP[31:0] = INT (TEMP[31:0] >> 14) + 1
DEST[79:64] = TEMP[16:1]
TEMP[31:0] = A[95:80] x B[95:80]
TEMP[31:0] = INT (TEMP[31:0] >> 14) + 1
DEST[95:80] = TEMP[16:1]
TEMP[31:0] = A[111:96] x B[111:96]
TEMP[31:0] = INT (TEMP[31:0] >> 14) + 1
DEST[111:96] = TEMP[16:1]
TEMP[31:0] = A[127:112] x B[127:112]
TEMP[31:0] = INT (TEMP[31:0] >>14) + 1
DEST[127:112] = TEMP[16:1]
747
749
エンド

フロントページの続き

(72)発明者　デリン　シー　ウォルターズ
アメリカ合衆国　テキサス州　７８７５８　オースティン　メドウファイア　ドライヴ　１１９３１
(72)発明者　ジョナサン　ジェイ　タイラー
アメリカ合衆国　テキサス州　７８７２８　オースティン　ドリア　ドライヴ　１４９２１

審査官　緑川　隆

(56)参考文献　特表平１１－５００５４７（ＪＰ，Ａ）
特表２００２－５２７８０８（ＪＰ，Ａ）
特表２００１－５１６９１６（ＪＰ，Ａ）
特開平０３－２６８０２４（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　　７／３８
Ｇ０６Ｆ　　７／４９６
Ｇ０６Ｆ　　９／３０５
Ｇ０６Ｆ　　９／３１５