(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4457084号
(P4457084)
(45)発行日　平成22年4月28日(2010.4.28)　(24)登録日　平成22年2月12日(2010.2.12)

(51)Int.Cl.　F I
G06F 7/38 (2006.01)　G06F 7/38 B

請求項の数 2　(全 13 頁)

(21)出願番号　特願2006-105931(P2006-105931)
(22)出願日　平成18年4月7日(2006.4.7)
(65)公開番号　特開2007-280082(P2007-280082A)
(43)公開日　平成19年10月25日(2007.10.25)
審査請求日　平成20年9月19日(2008.9.19)

(73)特許権者　308033711
ＯＫＩセミコンダクタ株式会社
東京都八王子市東浅川町５５０番地１
(74)代理人　100086807
弁理士　柿本　恭成
(72)発明者　内田　航
東京都港区虎ノ門１丁目７番１２号　沖電気工業株式会社内

審査官　緑川　隆

最終頁に続く

(54)【発明の名称】演算装置

(57)【特許請求の範囲】
【請求項１】
それぞれｚビット（但し、ｚ≧２）からなる乗数及び被乗数が入力され、前記乗数及び前記被乗数のうちの各上位ｚ／２ビットの特定領域が何ビット使用されているか否かを判定し、前記特定領域がｘビット（但し、０≦ｘ≦ｚ／２）使用されていれば、前記乗数及び前記被乗数における上位（ｚ／２－ｘ）ビットと下位ｘビットとを捨ててｚ／２ビットにそれぞれ丸める丸め処理手段と、
前記捨てられた各ビット数の情報をそれぞれ記憶する記憶手段と、
前記丸め処理手段により丸められた前記乗数と前記被乗数とを乗算して乗算結果を出力するｚ／２ビットの乗算器と、
前記記憶手段に記憶された前記ビット数の情報に基づき前記乗算結果をシフトして桁を調整する桁調整手段と、
を有することを特徴とする演算装置。
【請求項２】
それぞれｚビット（但し、ｚ≧２）からなる複数の乗数及び被乗数が入力され、前記複数の乗数及び被乗数のうちの各上位ｚ／２ビットの特定領域が何ビット使用されているか否かを検出して、使用されているビット数の最大値ｘ（但し、０≦ｘ≦ｚ／２）を求め、前記乗数及び前記被乗数における上位（ｚ／２－ｘ）ビットと下位ｘビットとを捨ててｚ／２ビットにそれぞれ丸める丸め処理手段と、
前記捨てられたビット数の情報を記憶する記憶手段と、

前記丸め処理手段により丸められた前記乗数と前記被乗数とを乗算して乗算結果を出力するｚ／２ビットの乗算器と、
前記記憶手段に記憶された前記ビット数の情報に基づき前記乗算結果をシフトして桁を調整する桁調整手段と、
を有することを特徴とする演算装置。

【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、例えば、固定小数点演算において低精度の乗算器を使用して演算結果の精度を向上させると共に演算を高速化するための丸め演算方法及びこの演算装置に関するものである。
【背景技術】
【０００２】
従来、例えば、音声圧縮技術であるＭＰ３（MPEG-1 Audio Layer 3）をデコード（解読）するデコーダでは、高速乗算処理を必要とする。
【０００３】
図２は、従来のＭＰ３ファイルをデコードする処理のフロー（流れ）を示す図である。
ＭＰ３のデコードにおいては、図示しないデコーダは、１７ビット（以下「bit」という。）のＭＰ３ファイル１から読み込んだデータを基に（ステップＳ１）、物理量として意味のある行列データを生成する逆量子化処理を行う（ステップＳ２）。デコーダは、逆量子化して得られた１７bitの各行列の要素を、図示しない高速な１６bitの乗算器に入力するために（ステップＳ３）、１６bitデータ２に丸める（即ち、０捨１入）必要がある（ステップＳ４）。そのためには、ＭＰ３ファイル１の最上位bit（以下「ＭＳＢ」という。）又は最下位bit（以下「ＬＳＢ」という。）を捨てなければならない。丸められたデータは、乗算器で演算され（ステップＳ５）、デコーダから出力される（ステップＳ６）。
【０００４】
ところで、先に述べたデータを１６bitに丸める処理（ステップＳ４）において、ＬＳＢを捨てるとデータ精度が悪化する。これに対し、ＭＳＢを捨てるとデータがＭＳＢまでフルに使用している場合、データが意図しない数値に丸められ、その結果、デコードした音が歪む恐れがある。
【０００５】
このような丸めの問題を解決するための手法が、従来の特許文献１で提案されている。
【０００６】
【特許文献１】米国特許第６，３６０，２０４Ｂ１
【０００７】
図３は、従来の特許文献１に記載された丸め演算方法を示す図である。
この丸め演算方法では、以下のように丸め処理を工夫することによって、オーディオデコーダの精度を向上させている。
【０００８】
乗数３及び被乗数４がそれぞれｓbitのシグナルプロセッサ（Digital signal processor、以下「ＤＳＰ」という。）によるオーディオデータを使用した乗算結果５は、最大２ｓ bitになるため、これを次のような手順でｓbitに丸める。
【０００９】
先ず、乗算結果５の上位ｓbit又は下位ｓbitのどちらを確保するかを選択する。一般に、上位bitの丸め処理では丸め精度が低く、下位bitの丸め処理では丸め精度が高い。これらの選択方法は任意である。次に、上位ｓbitを確保する場合は、データの飽和の有無（即ち、ＭＳＢまで使用しているか否か）を確認し（ステップＳ１０）、飽和がなければ丸め処理を行い（ステップＳ１１）、飽和していれば丸めない。
【発明の開示】

【発明が解決しようとする課題】
【００１０】
従来の丸め演算方法により、ｓbitのＤＳＰに実装したオーディオデコーダの精度を向上させることが可能になるが、図３に示すように、選択可能な精度が上位bitを確保する場合と下位bitを確保する場合、２通りに限定される問題があり、アプリケーション（応用）もオーディオデコーダに限定されている。
【００１１】
つまり、従来の丸め演算方法あるいは演算装置では、例えば、汎用のマイクロプロセッサ（以下「ＭＣＵ」という。）において、bit数の少ない低精度の乗算器を用いて乗算を高速化すると同時に、演算データの精度を柔軟に選択して確保することが難しく、しかも、対象アプリケーションをオーディオデコーダに限定せずに適用可能にすることが困難であった。
【課題を解決するための手段】
【００１６】
請求項１に係る発明の演算装置では、ｚbit（但し、ｚ≧２）からなる入力データのうちの上位ｚ／２bitの特定領域が何ビット使用されているか否かを判定し、前記特定領域がｘbit（但し、０≦ｘ≦ｚ／２）使用されていれば、前記入力データにおける上位（ｚ／２－ｘ）bitと下位ｘbitとを捨ててｚ／２bitに丸める。
【００１７】
請求項１に係る発明の演算装置では、丸め処理手段と、記憶手段と、ｚ／２bitの乗算器と、桁調整手段とを有している。
【００１８】
前記丸め処理手段は、それぞれｚbit（但し、ｚ≧２）からなる乗数及び被乗数が入力され、前記乗数及び前記被乗数のうちの各上位ｚ／２bitの特定領域が何bit使用されているか否かを判定し、前記特定領域がｘbit（但し、０≦ｘ≦ｚ／２）使用されていれば、前記乗数及び前記被乗数における上位（ｚ／２－ｘ）bitと下位ｘbitとを捨ててｚ／２bitにそれぞれ丸める。前記記憶手段は、前記捨てられた各bit数の情報をそれぞれ記憶する。前記乗算器は、前記丸め処理手段により丸められた前記乗数と前記被乗数とを乗算して乗算結果を出力する。更に、前記桁調整手段は、前記記憶手段に記憶された前記ビット数の情報に基づき前記乗算結果をシフトして桁を調整する。
【００１９】
このように、請求項１の演算装置では、丸め込み処理において、確保する下位ビットフィールド（複数bit）の範囲を条件によって動的に変化させることにより、高速な低精度乗算器を使用して演算処理を高速化し、且つ、演算の出力精度を確保している。
【００２０】
請求項２に係る発明の演算装置では、それぞれｚbit（但し、ｚ≧２）からなる複数の入力データにおいて、前記各入力データのうちの上位ｚ／２bitの特定領域が何ビット使用されているか否かを検出して、使用されているbit数の最大値ｘ（但し、０≦ｘ≦ｚ／２）を求め、前記各入力データにおける上位（ｚ／２－ｘ）bitと下位ｘbitとを捨ててｚ／２bitにそれぞれ丸める。
【００２１】
この請求項２に係る発明の演算装置では、丸め処理手段と、記憶手段と、ｚ／２bitの乗算器と、桁調整手段とを有している。
【００２２】
前記丸め処理手段は、それぞれｚbit（但し、ｚ≧２）からなる複数の乗数及び被乗数が入力され、前記複数の乗数及び被乗数のうちの各上位ｚ／２bitの特定領域が何bit使用されているか否かを検出して、使用されているbit数の最大値ｘ（但し、０≦ｘ≦ｚ／２）を求め、前記乗数及び前記被乗数における上位（ｚ／２－ｘ）bitと下位ｘbitとを捨ててｚ／２bitにそれぞれ丸める。前記記憶手段は、前記捨てられたbit数の情報を記憶する。前記乗算器は、前記丸め処理手段により丸められた前記乗数と前記被乗数とを乗算して

乗算結果を出力する。更に、前記桁調整手段は、前記記憶手段に記憶された前記bit数の情報に基づき前記乗算結果をシフトして桁を調整する。

【００２３】

このように、請求項３、６の丸め演算方法あるいは演算装置では、丸め込み処理において、丸め込むbit数を計算単位で保持することによって、演算の出力精度を確保して記憶容量を削減している。

【発明の効果】

【００２４】

請求項１に係る発明によれば、次の（ｉ）～（iii）のような効果がある。

【００２５】

（ｉ）　丸め処理を工夫しているので、演算結果の精度を確保することが可能である。

【００２６】

（ii）　所望（ｚ／２bit）の乗算器を用いることができるため、例えば、ｚ／２bit積和演算を高速に実行できるプロセッサでは、高速化することが可能になる。

【００２７】

（iii）　確保するbit数を動的に変化させることにより、bit数の許す限りの精度を確保できるため、乗算器による演算結果精度を改善することが可能になる。

【００２９】

請求項２に係る発明によれば、請求項１の効果（ｉ）、（ii）と同様の効果があり、更に、計算単位毎にシフト情報を一括して管理するため、請求項１に係る発明と比較して、メモリ容量を削減することが可能になる。

【発明を実施するための最良の形態】

【００３０】

固定小数点演算における丸め演算方法では、ｚbit（但し、ｚ≧２）からなる入力データのうちの上位ｎbit（但し、ｚ＞ｎ≧２）の特定領域が使用されているか否かを判定し、前記特定領域が使用されていなければ、前記入力データにおける前記上位ｎbitと下位（ｚ／２－ｎ）bitとを捨ててｚ／２bit値に丸め、前記特定領域が使用されていれば、前記入力データにおける下位ｚ／２bitを捨ててｚ／２bitに丸める。

【実施例１】

【００３１】

図１（１）、（２）は、本発明の実施例１を示す演算装置の説明図であり、同（１）は、概略の構成図、及び同図（２）は、処理内容を示す図である。

【００３２】

この演算装置は、例えば、ＭＰ３のデコードのアプリケーションに使用される固定小数点演算の乗算を行う装置であり、複数（Ｋ）のｚbit（例えば、３２bit）からなる入力データＩＮ１～ＩＮＫの集合を入力する丸め処理手段（例えば、丸め処理部）１０を有している。丸め処理部１０は、各３２bitの入力データＩＮ１～ＩＮＫのうちの上位ｎbit（固定値）の特定領域Ａの使用状態によって、確保するビットフィールドＣを選択してｚ／２（例えば、１６bit）に丸め込む機能を有している。

【００３３】

即ち、丸め処理部１０は、各入力データＩＮ１～ＩＮＫについて、上位ｎbitの特定領域Ａが使用されているか否かを判定し、特定領域Ａが使用されていなければ（図１（２）の（ａ）の場合）、この上位ｎbitの特定領域Ａと、残りの（３２－ｎ）bitの非特定領域Ｂのうちの下位（ｚ／２－ｎ＝１６－ｎ）bitとを捨て、確保するビットフィールドＣを選択して１６bit値に丸め、特定領域Ａが使用されていれば（図１（２）の（ｂ）の場合）、非特定領域Ｂのうちの下位１６bitを捨て、確保するビットフィールドＣを選択して１６bit値に丸める機能を有している。

【００３４】

このような丸め処理部１０には、記憶手段（例えば、メモリ）２０、及びｚ／２bit（例えば、１８bit）の乗算器３０が接続され、更に、そのメモリ２０及び乗算器３０に、

桁調整手段（例えば、桁調整部）４０が接続されている。

【００３５】

メモリ２０は、丸め込む際に切り捨てた各bit数の情報（即ち、bit幅の情報）をシフト情報SHIFTとして保存するための複数（Ｋ）のメモリ領域２１－１～２１－Ｋを有している。１８bitの乗算器３０は、丸められた複数の１６bitのデータＤ１０－１～Ｄ１０－Ｋのうちから、乗数となる１６bitのデータと被乗数となる１６bitのデータとをそれぞれ乗算して、各３２bitの乗算結果Ｄ３０を桁調整部４０へ出力する回路である。桁調整部４０は、各メモリ領域２１－１～２１－Ｋに保存された各シフト情報SHIFTに基づき、各乗算結果Ｄ３０をそれぞれシフトして桁を調整するものである。

【００３６】

丸め処理部１０及び桁調整部４０は、算術論理ユニット（以下「ＡＬＵ」という。）あるいはシフタ等で構成されている。

【００３７】

（実施例１の丸め演算方法を含む乗算処理）

図４は、図１（２）の（ａ）、（ｂ）の場合の丸め演算方法を示す説明図である。図５は、図１の丸め演算方法を含む乗算処理を示すフローチャートである。

【００３８】

３２bitの入力データＩＮ１～ＩＮＫの集合に対する乗算処理は、図５のフローチャートに従って以下のように実行される。

【００３９】

乗算処理が開始されて（ステップＳ２０）、３２bitの入力データＩＮ１～ＩＮＫの集合が丸め処理部１０に入力されると（ステップＳ２１）、この丸め処理部１０では、ある入力データ（例えば、ＩＮ１，ＩＮ２）が上位ｎbit（例えば、3bit）の特定領域Ａを使用しているか否かを判定し（ステップＳ２２）、この判定結果に基づいて丸め処理を行う（ステップＳ２３－１，Ｓ２３－２）。

【００４０】

例えば、図４の（ａ）の場合のように、入力データＩＮ１が上位ｎbit（＝3bit）の特定領域Ａを使用していなければ（“０００”）、この上位ｎbit（＝3bit）と非特定領域Ｂの下位（１６－ｎ）bit（＝１３bit）とを捨て（即ち、ＭＳＢから“０”を挿入して（１６－ｎ）＝１３bitだけ右にシフトし）、確保するビットフィールドＣを選択して１６bit値に丸める（ステップＳ２３－１）。

【００４１】

これに対し、図４の（ｂ）の場合のように、入力データＩＮ２が上位ｎbit（＝3bit）の特定領域Ａを使用していれば（“１０１”）、非特定領域Ｂの下位１６bitを捨て（即ち、ＭＳＢから“０”を挿入して１６bitだけ右にシフトし）、確保するビットフィールドＣを選択して１６bit値に丸める（ステップＳ２３－２）。

【００４２】

図４の（ａ）の丸め処理の結果、捨てられた非特定領域Ｂの下位（１６－ｎ）＝１３bitのシフト情報SHIFTをメモリ領域２１－１に格納する（ステップＳ２４－１）。又、図４の（ｂ）の丸め処理の結果、捨てられた非特定領域Ｂの下位１６bitのシフト情報SHIFTをメモリ領域２１－２に格納する（ステップＳ２４－２）。

【００４３】

他の入力データＩＮ３～ＩＮＫについても、同様に丸め処理する。メモリ領域２１－１～２１－Ｋは、入力データＩＮ１～ＩＮＫの個数分用意されているので、各入力データＩＮ１～ＩＮＫに対応するシフト情報SHIFTは、各メモリ領域２０－１～２０－Ｋに個別に格納される。

【００４４】

丸められた各１６bitのデータＤ１０－１～Ｄ１０－Ｋは、乗算器３０で乗算され（ステップＳ２５）、３２bitの乗算結果Ｄ３０が桁調整部４０へ送られる。桁調整部４０では、メモリ領域２１－１～２１－ｋにそれぞれ格納されたシフト情報SHIFTに基づき、乗

算結果Ｄ３０を左にシフトして桁を調整し（ステップＳ２６）、出力データＯＵＴを出力して処理を終了する（ステップＳ２７）。

【００４５】

（実施例１の効果）

本実施例１によれば、次の（１）～（４）のような効果がある。

【００４６】

（１）　丸め処理を工夫しているので、演算結果の精度を確保することが可能である。

【００４７】

（２）　１６bitの乗算器３０を用いることができるため、例えば、１６bit積和演算を高速に実行できるプロセッサでは、高速化することが可能になる。

【００４８】

（３）　確保するビットフィールドＣのbit数（ｎbit）を調整することにより、アプリケーションに合わせた精度を確保することが可能である。

【００４９】

（４）　図６（１）、（２）は、本発明の実施例と従来との演算精度を比較した図であり、同図（１）は比較したシミュレーション結果を示す図、及び同図（２）は同図（１）の演算精度の求め方を示す図である。

【００５０】

図６（２）に示すように、演算精度（即ち、丸めの誤差）を求める場合、複数個（例えば、１２８個）の入力データＩＮ１～ＩＮＫを演算装置に入力し、演算結果の真の値５０と丸めた値５１との誤差５２を求め、この誤差５２の平均値を求めれば良い。誤差５２の平均値は、下記のＲＭＳ（実効値ＶＲＭＳ）で表すことができる。

【００５１】

【数１】

$$V_{RMS} = \sqrt{\frac{1}{T} \cdot \int_{\Delta t = T} V(t)^2 \cdot dt}$$

但し、 t : 入力データのインデックス

v(t) : 誤差

T : 入力データ数

【００５２】

図６（１）のシミュレーション結果は、上位１６bitを確保した場合（従来の曲線５３）の精度（ＲＭＳ従来の直線５４）と本発明の実施例１を使用した場合（実施例１の曲線５５）の精度（ＲＭＳ実施例の直線５６）とを比較したものである。例として、１２８サンプルの３２bit入力値をそれぞれ１６bit値に丸め、３２bit入力値との誤差を示している。このシミュレーション結果によると、ＲＭＳ従来の直線５４とＲＭＳ実施例の直線５６から、従来の演算精度よりも本実施例１の演算精度の方が誤差が小さく精度が高いことが分かる。

【実施例２】

【００５３】

（実施例２の構成）

図７は、本発明の実施例２における演算装置の処理内容を示す図であり、実施例１を示す図１（２）中の要素と共通の要素には共通の符号が付されている。

【００５４】

本実施例２の演算装置は、実施例１とは処理内容の異なる丸め処理手段（例えば、丸め処理部）１０－１と、実施例１と同様の記憶手段（例えば、メモリ）２０、ｚ／2bit（

例えば、１８bit）の乗算器３０、及び桁調整手段（例えば、桁調整部）４０とにより構成されている。

【００５５】

丸め処理部１０－１は、各３２bitの入力データＩＮ１～ＩＮＫのうちの上位ｚ／２bitの特定領域Ａの使用状態によって、確保するビットフィールドＣをデータ個別に動的に変化させてｚ／２（例えば、１６bit）に丸め込む機能を有している。即ち、この丸め処理部１０－１は、各入力データＩＮ１～ＩＮＫについて、上位ｎbitの特定領域Ａが何bit使用されているか否かを判定し、ｘbit（０≦ｘ≦ｚ／２）使用されていれば、特定領域Ａの上位（ｚ／２－ｘ）bitと、非特定領域Ｂの下位ｘbitとを捨て、確保するビットフィールドＣを選択して１６bit値に丸める機能を有している。

【００５６】

実施例１と同様に、丸め込む際に切り捨てたビット幅の情報はシフト情報SHIFTとして、メモリ２０内に用意されたデータの個数分のメモリ領域２１－１～２１－Ｋに保存され、乗算器３０で乗算された乗算結果の桁を、桁調整部４０にてシフトして調整するために使用される構成になっている。

【００５７】

（実施例２の丸め演算方法を含む乗算処理）

図８は、図７の丸め演算方法を含む乗算処理を示すフローチャートであり、実施例１を示す図５中の要素と共通の要素には共通の符号が付されている。

【００５８】

３２bitの入力データＩＮ１～ＩＮＫの集合に対する乗算処理は、図８のフローチャートに従って以下のように実行される。

【００５９】

乗算処理が開始されて（ステップＳ２０）、３２bitの入力データＩＮ１～ＩＮＫの集合が丸め処理部１０－１に入力されると（ステップＳ２１）、この丸め処理部１０－１では、ある入力データ（例えば、ＩＮ１，ＩＮ２）が上位ｚ／２bit（例えば、１６bit）の特定領域Ａを何bit（ｘbit）使用しているか否かを判定する（ステップＳ３２）。入力データ（例えば、ＩＮ１，ＩＮ２）が特定領域Ａをｘbit（０≦ｘ≦１６）使用しているとすると、特定領域Ａの上位（１６－ｘ）bitと、非特定領域Ｂの下位ｘbitとを捨て、確保するビットフィールドＣを選択して１６bit値に丸める（ステップＳ３３）。

【００６０】

例えば、入力データＩＮ１が特定領域Ａ中のｘ（＝３）bitを使用している場合、この特定領域Ａの上位（１６－ｘ＝１３）bitと、非特定領域Ｂの下位ｘ（＝３）bitとを捨てて１６bit値に丸める。又、入力データＩＮ２が特定領域Ａ中のｘ（＝５）bitを使用している場合、この特定領域Ａの上位（１６－ｘ＝１１）bitと、非特定領域Ｂの下位ｘ（＝５）bitとを捨てて１６bit値に丸める。

【００６１】

丸め処理の結果、捨てられた非特定領域Ｂの下位ｘbitのシフト情報SHIFTをメモリ領域２１－１，２１－２に格納する（ステップＳ３４）。他の入力データＩＮ３～ＩＮＫについても、同様に丸め処理する。実施例１と同様に、メモリ領域２１－１～２１－Ｋは、入力データＩＮ１～ＩＮＫの個数分用意されているので、各入力データＩＮ１～ＩＮＫに対応するシフト情報SHIFTは、各メモリ領域２１－１～２１－Ｋに個別に格納される。

【００６２】

丸められた各１６bitのデータＤ１０－１～Ｄ１０－Ｋは、実施例１と同様に、乗算器３０で乗算され（ステップＳ２５）、３２bitの乗算結果Ｄ３０が桁調整部４０へ送られる。桁調整部４０では、メモリ領域２０－１～２０－ｋにそれぞれ格納されたシフト情報SHIFTに基づき、乗算結果Ｄ３０を左にシフトして桁を調整し（ステップＳ２６）、処理を終了する（ステップＳ２７）。

【００６３】

（実施例２の効果）

本実施例２によれば、実施例１の効果（１）、（２）と同様の効果があり、更に、次の（４）のような効果がある。

【００６４】

（４） 確保するbit数を動的に変化させることにより、bit数の許す限りの精度を確保できるため、実施例１と比較して乗算器３０による演算結果精度を改善することが可能になる。これに関して、図６と同様のシミュレーション結果によれば、実施例１よりも精度が向上できた。

【実施例３】

【００６５】

（実施例３の構成）

図９は、本発明の実施例３における演算装置の処理内容を示す図であり、実施例１を示す図１（２）中の要素と共通の要素には共通の符号が付されている。

【００６６】

本実施例３の演算装置は、実施例１とは処理内容の異なる丸め処理手段（例えば、丸め処理部）１０－２と、実施例１とは記憶容量の異なる記憶手段（例えば、メモリ）２０－１と、実施例１と同様のｚ／２bit（例えば、１８bit）の乗算器３０、及び桁調整手段（例えば、桁調整部）４０とにより構成されている。

【００６７】

丸め処理部１０－２は、各ｚbit（例えば、３２bit）の入力データＩＮ１～ＩＮＫの上位ｚ／２bit（例えば、１６bit）の特定領域Ａの使用状態によって、確保するビットフィールドＣを動的に変化させてｚ／２（例えば、１６bit）に丸め込む機能を有している。即ち、この丸め処理部１０－２は、複数の入力データデータＩＮ１～ＩＮＫのうちの各上位ｎbit（例えば、１６bit）の特定領域Ａが何ビット使用されているか否かを検出して、使用されているビット数の最大値ｘ（但し、０≦ｘ≦ｚ／２）を求め、入力データＩＮ１～ＩＮＫにおける上位（ｚ／２－ｘ）bitと下位ｘbitとを捨ててｚ／２bitにそれぞれ丸める機能を有している。

【００６８】

丸め込む際に切り捨てたビット幅の情報はシフト情報SHIFTとして、メモリ２０－１内のメモリ領域２１－１に保存される。シフト情報SHIFTは、データの計算単位（データ集合）毎に保持され、乗算器３０で乗算された乗算結果の桁を、桁調整部４０にてシフトして調整するために使用される構成になっている。

【００６９】

（実施例３の丸め演算方法を含む乗算処理）

図１０は、図９の丸め演算方法を含む乗算処理を示すフローチャートであり、実施例１を示す図５中の要素と共通の要素には共通の符号が付されている。

【００７０】

３２bitの入力データＩＮ１～ＩＮＫの集合に対する乗算処理は、図１０のフローチャートに従って以下のように実行される。

【００７１】

乗算処理が開始されて（ステップＳ２０）、３２bitの入力データＩＮ１～ＩＮＫの集合が丸め処理部１０－２に入力されると（ステップＳ２１）、この丸め処理部１０－２では、データ集合のｚbit（例えば、３２bit）長の各入力データＩＮ１～ＩＮＫが、上位ｚ／２bit（例えば、１６bit）の特定領域Ａを何bit使用しているかを検出する（ステップＳ４２）。使用しているbit数の最大値がｘbit（０≦ｘ≦１６）であるとすると、各入力データＩＮ１～ＩＮＫの上位（１６－ｘ）bitと下位ｘbitを捨て、確保するビットフィールドＣを選択して１６bit値に丸める（ステップＳ４３）。

【００７２】

例えば、入力データＩＮ１が特定領域Ａ中のｘ（＝４）bitを使用し、入力データＩＮ２が5bitを使用し、入力データＩＮＫが3bitを使用している場合、使用しているbit数の最大値ｘが5bitとなる。そこで、各入力データＩＮ１～ＩＮＫの上位（１６－ｘ＝１

１）　bitと下位ｘ（＝５）bitを捨てて１６bit値に丸める。

【００７３】

丸め処理の結果、各入力データＩＮ１～ＩＮＫにおいて捨てられた非特定領域Ｂの下位ｘ（＝５）bitのシフト情報SHIFTをメモリ領域２１－１に格納する（ステップＳ４４）。捨てられるbit数（＝５）は、データ集合の各入力データＩＮ１～ＩＮＫにおいて共通である。このため、メモリ領域２１－１は１つ用意し、各データ共通に使用する。

【００７４】

丸められた各１６bitのデータＤ１０－１～Ｄ１０－Ｋは、実施例１と同様に、乗算器３０で乗算され（ステップＳ２５）、３２bitの乗算結果Ｄ３０が桁調整部４０へ送られる。桁調整部４０では、メモリ領域２１－１に格納されたシフト情報SHIFTに基づき、乗算結果Ｄ３０を左にシフトして桁を調整し（ステップＳ２６）、処理を終了する（ステップＳ２７）。

【００７５】

（実施例３の効果）

本実施例３によれば、実施例１の効果（１）、（２）と同様の効果があり、更に、次の（５）のような効果がある。

【００７６】

（５）　計算単位毎にシフト情報SHIFTを一括して管理するため、実施例２と比較して、メモリ容量を削減することが可能になる。これに関して、図６と同様のシミュレーション結果によれば、実施例１と実施例２の中間の精度が確保できた。

【００７７】

（変形例）

本発明は実施例１～３に限定されず、種々の利用形態や変形が可能である。この利用形態や変形例としては、例えば、次の（I）、（II）のようなものがある。

【００７８】

（I）　演算装置を構成する丸め処理部１０，１０－１，１０－２、乗算器３０、及び桁調整部４０のbit数は任意であり、更に、これらの演算装置を図示以外の他の構成に変更しても良い。

【００７９】

（II）　実施例１～３では、ＭＰ３のデコードを例に挙げたが、本発明は、乗算器３０を使用する演算の高速化等の目的でデータを丸める必要がある種々のアプリケーションにおいて精度を確保することを可能にする。

【図面の簡単な説明】

【００８０】

【図１】本発明の実施例１を示す演算装置の説明図である。

【図２】従来のＭＰ３ファイルをデコードする処理のフローを示す図である。

【図３】従来の特許文献１に記載された丸め演算方法を示す図である。

【図４】図１（２）の（ａ）、（ｂ）の場合の丸め演算方法を示す説明図である。

【図５】図１の丸め演算方法を含む乗算処理を示すフローチャートである。

【図６】本発明の実施例と従来との演算精度を比較した図である。

【図７】本発明の実施例２における演算装置の処理内容を示す図である。

【図８】図７の丸め演算方法を含む乗算処理を示すフローチャートである。

【図９】本発明の実施例３における演算装置の処理内容を示す図である。

【図１０】図９の丸め演算方法を含む乗算処理を示すフローチャートである。

【符号の説明】

【００８１】

１０，１０－１，１０－２　　丸め処理部

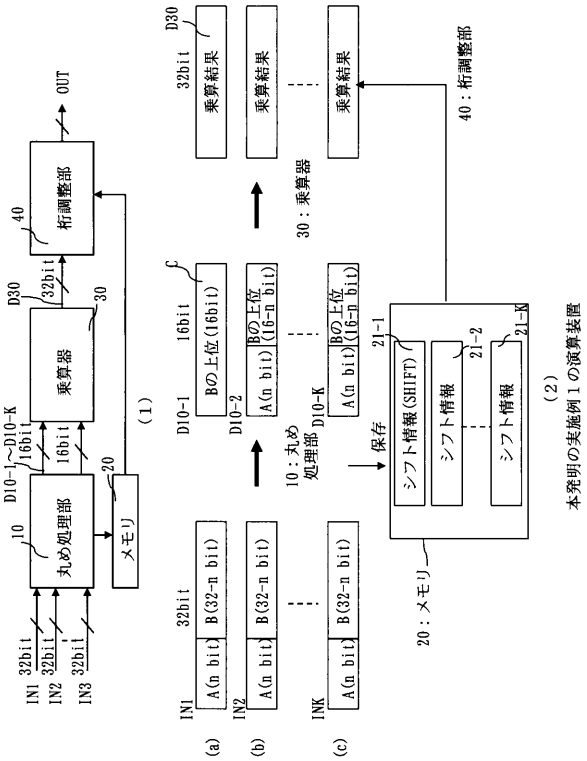
２０，２０－１　　メモリ

２１－１～２１－Ｋ　　メモリ領域

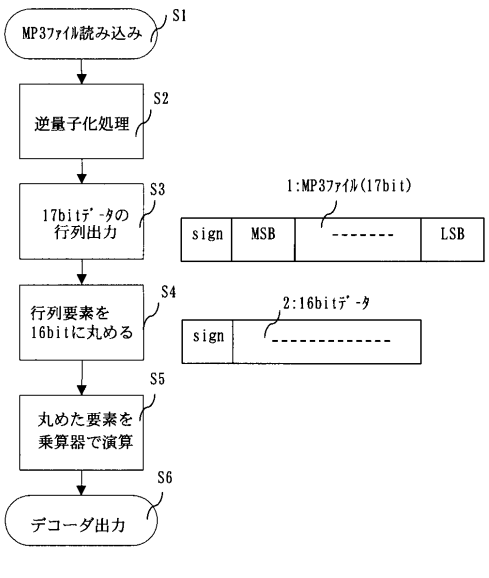
３０　　乗算器

４０　　桁調整部

【図１】

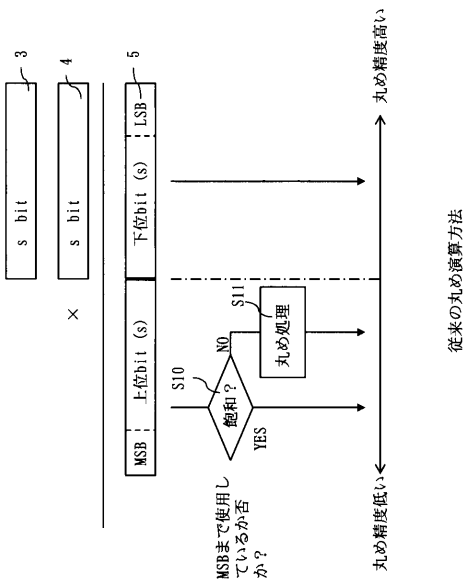


【図２】



MP3ファイルデコード処理フロー

【図３】



【図４】

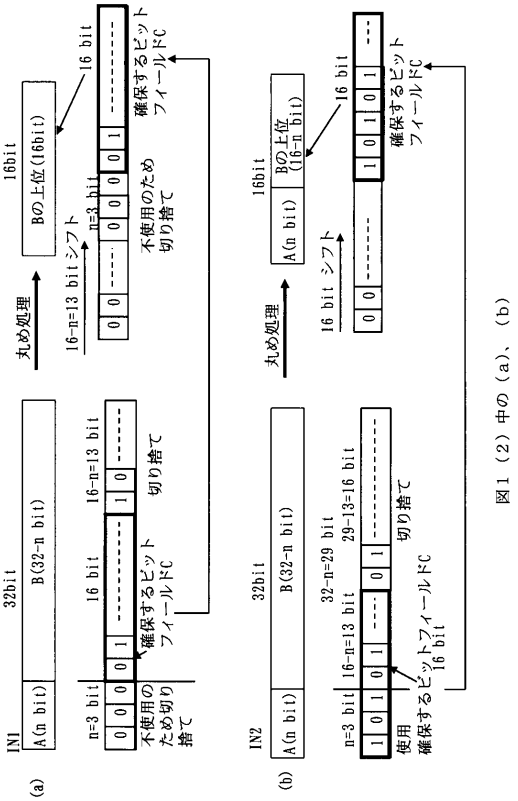


図１（２）中の（ａ）、（ｂ）

【図５】

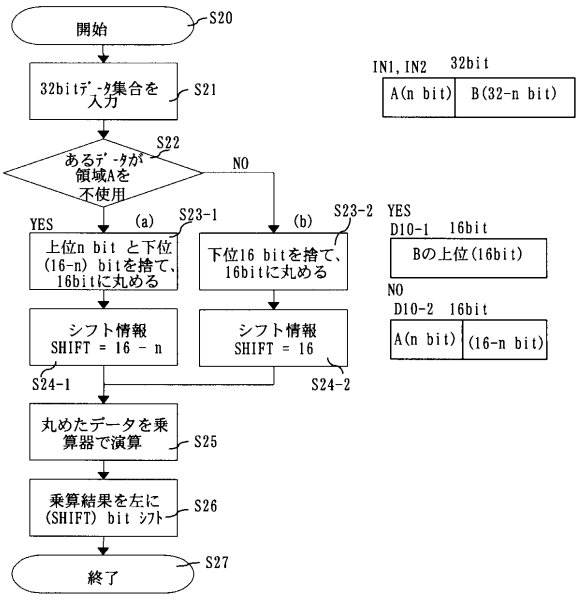
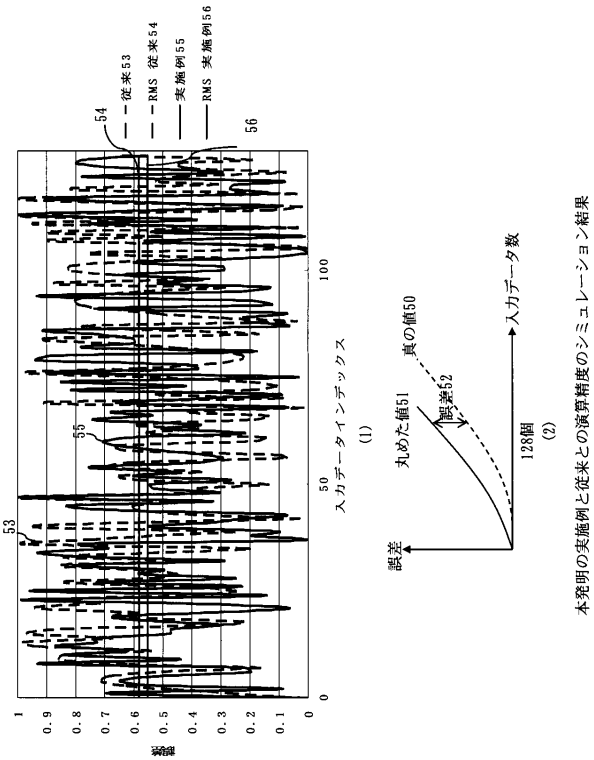


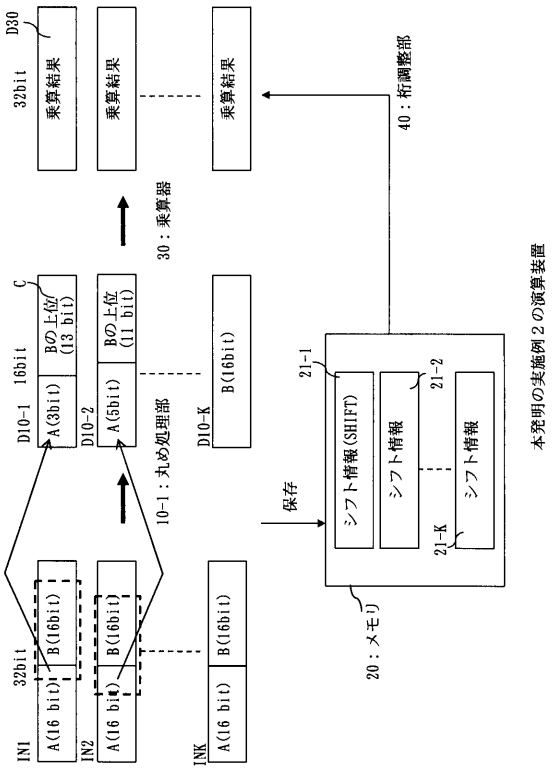
図１のフロー

【図６】



本発明の実施例と従来との演算精度のシミュレーション結果

【図７】



【図８】

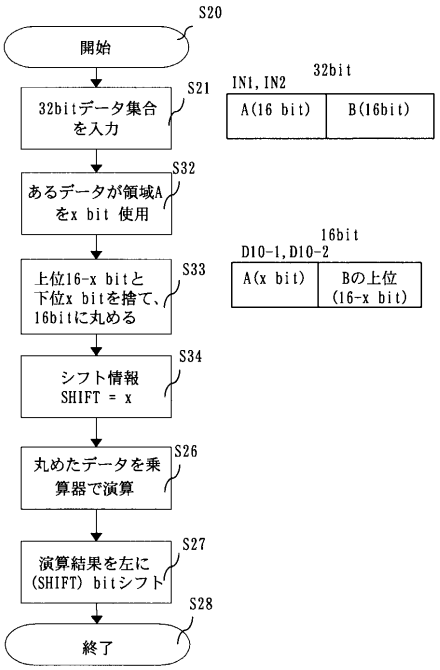
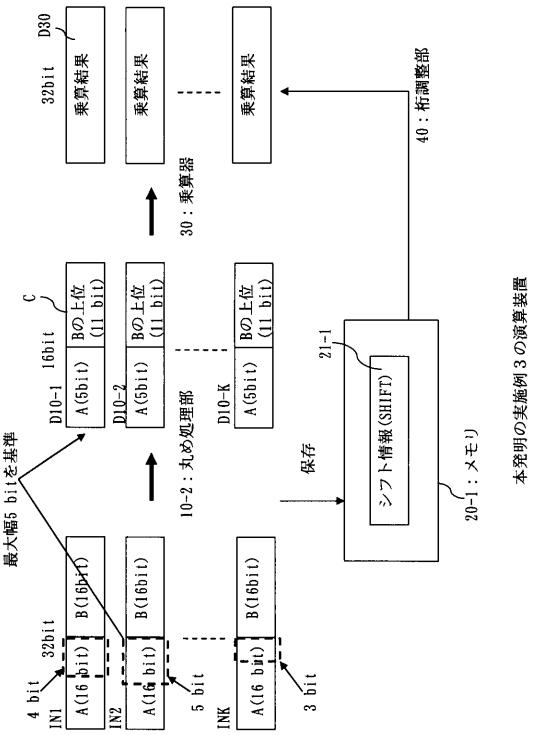


図７のフロー

【図９】



【図１０】

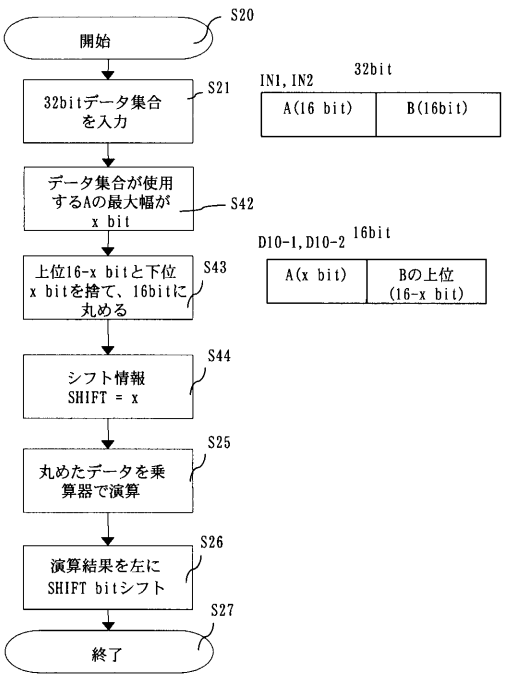


図９のフロー

フロントページの続き

(56)参考文献 特開平０６－０３５６７２（ＪＰ，Ａ）
特開平０５－０４０６０５（ＪＰ，Ａ）
特開２００２－３０４２８８（ＪＰ，Ａ）
国際公開第９９／０６６４２３（WO，Ａ１）
特開平０５－０４０６０６（ＪＰ，Ａ）
特開２００３－０６７１８２（ＪＰ，Ａ）
特開平１０－０４００７３（ＪＰ，Ａ）
特開平１１－２１９４３２（ＪＰ，Ａ）
特開平０５－０７３２６６（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　　７／３８