(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3753275号
(P3753275)
(45)発行日　平成18年3月8日(2006.3.8)　(24)登録日　平成17年12月22日(2005.12.22)

(51)Int.Cl.　F I
G06F 7/00 (2006.01)　G06F 7/00 101G

請求項の数 3　(全 21 頁)

(21)出願番号　特願平9-24644
(22)出願日　平成9年2月7日(1997.2.7)
(65)公開番号　特開平9-244874
(43)公開日　平成9年9月19日(1997.9.19)
審査請求日　平成16年2月5日(2004.2.5)
(31)優先権主張番号　608,798
(32)優先日　平成8年2月29日(1996.2.29)
(33)優先権主張国　米国(US)

(73)特許権者　398038580
ヒューレット・パッカード・カンパニー
HEWLETT-PACKARD COMPANY
アメリカ合衆国カリフォルニア州パロアルト　ハノーバー・ストリート　3000
(74)代理人　100081721
弁理士　岡田　次生
(72)発明者　ロバート・エイチ・ミラー，ジュニア
アメリカ合衆国80537コロラド州ラブランド、ダブリュー・5ストリート　431

最終頁に続く

(54)【発明の名称】最上位有効ビット位置予測方法

(57)【特許請求の範囲】
【請求項１】
２つのｎビットのオリジナル・オペランドの２の補数の２進和の最上位有効ビット位置を予測する方法であって、
（ａ）前記２つのｎビットのオリジナル・オペランドを２つの(ｎ＋１)ビット形式に非正規化するステップと、
（ｂ）２つの(ｎ＋１)ビットで表される値のそれぞれのビット位置に記号を作成するステップであって、前記記号のそれぞれは、第１の記号、第２の記号、または第３の記号のうちの一つであって、前記第１の記号は２つのビットの加算がゼロであることに対応し、前記第２の記号は前記加算が１であることに対応し、前記第３の記号は前記加算が２であることに対応し、このようにして(ｎ＋１)個の記号のシーケンスを生成する、前記作成するステップと、
（ｃ）前記(ｎ＋１)個の記号を（ｎ－１）個の順序づけたトリプルに分類するステップであって、トリプルのそれぞれは隣接するトリプルに共通の２つの隣接する記号を有する、前記分類するステップと、
（ｄ）前記トリプルを検査するステップであって、少なくとも前記トリプルの数に等しい数の暫定ビットから構成される２進暫定ワードにおいて対応する暫定ビットを作成し、前記暫定ビットは前記トリプルの順序で対応し、前記暫定ビットの第１の暫定値を生成することは、検査されるトリプルに対応する、非正規化された２つのオペランドのビット位置の部分の和は、最上位からの上位２ビットが、そのままでも、繰り入れキャリを伴う場

合にも、異なるビット値となることを示し、前記暫定ビットの第２の暫定値を生成することは、異なるビット値とならないことを示す、前記検査するステップと、
（ｅ）前記２進暫定ワードの検査によって、前記第１の暫定値の最上位有効ビットのビット位置を求めるステップと、
（ｆ）求められた前記ビット位置をビット位置数Ｊで表すステップであって、前記Ｊは前記２進暫定ワードの最左端からのビット位置数である、前記表すステップと、
（ｇ）前記２つの(ｎ＋１)ビットで表される値を加算するステップであって、未調整和を作成する、前記加算するステップと、
（ｈ）前記未調整和を前記Ｊに等しい量だけ正規化するステップと、
を含む最上位有効ビット位置予測方法。
【請求項２】
（ｋ）前記第１の記号がＫを示し、第二の記号がＰを示し、第三の記号がＧを示し、
ＫＰＫ、ＫＰＰ、ＫＰＧ、ＧＰＫ、ＧＰＰ、またはＧＰＧの６つのトリプルのうちの１つがｎビット・オリジナル・オペランドの最上位ビットに関連付けられるとき、トリプルについて関連付けられた暫定ビット値について第１の暫定値を作成し、６つのトリプルのうちの１つが最上位ビットではないｎビット・オリジナル・オペランドのビットに関連付けられるとき、関連付けられた暫定ビットに第２の暫定値を作成するステップ、
をさらに含む請求項１記載の最上位有効ビット位置予測方法。
【請求項３】
（ｉ）Ｊが１ビット位置小さいことを判定するステップと、
（ｊ）Ｊに対応するシフトとは別に、前記未調整和にさらなるシフトを行うステップと、
をさらに含む請求項１記載の最上位有効ビット位置予測方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は、浮動小数点２進加算器に関するもので、特に該加算器において使用される最上位有効ビット位置予測の装置および方法に関するものである。
【０００２】
【従来の技術】
高性能浮動小数点２進加算器は、近代マイクロプロセッサの重要なコンポーネントである。高性能加算器は、旧式の腕力に任せた技術に頼るのではなく、ＰＫＧ符号化方式のような代替アプローチをしばしば使う。ＰＫＧ符号化方式は、迅速に和を計算するため、２つの２進加数を諸段階で使用される可能性のあるビット単位のキャリ情報に置き換える。加数のＰＫＧ表示は、２つのオリジナルのオペランドより多くのビットを必要とするが、和の計算速度の点で顕著な利益が得られる。ＰＫＧ方式であろうとそうでなかろうと、高性能浮動小数点加算機の重要な部分は、「最上位有効ビット位置予測器」(Leading Bit Anticipatorの頭文字をとってＬＢＡと呼称される)と呼ばれるエレメントである。
【０００３】
すべての浮動小数点演算の場合、演算の未調整結果は正規化されなければならないことが多い。すなわち、加算後仮数部の上位桁のゼロを削除するため有効ビットをシフトする必要があり(あるいは、負である２の補数の場合にも有効桁をシフトしてゼロを削除する必要があり)、そのシフト数に対応して指数部も調整される。ＬＢＡがない場合、演算後の未調整結果は、どのような正規化ステップが必要かを決定するため検査されなければならない。これが機能しないことがしばしばあるわけではないが、最終的解を得るまでに相当の時間を要する。ＬＢＡは、演算の和が計算される間に正規化のステップを決定するので、未調整演算結果が計算され次第追加の検査を行うことなく直ちに正規化を実行することができる。正規化ステップとは、正規化の際にゼロのビットを削除し、指数調整に必要なシフト桁数を決定することを含む。
【０００４】

既存の最上位有効ビット位置予測器(ＬＢＡ)は複雑で、その実施には多くのユニットを必要とする。ＬＢＡの１つの例が、IBM JOURNAL OF RESEARCH AND DEVELOPMENT, January 1990 (Vol. 34, No. 1) pages 59-70に"Design of the IBM RISC System/6000 floating-point execution unit(IBM RISC System/6000浮動小数点演算実行装置の設計)"と題するR. K. Montoya, E. Hokenek, S. L. Runyon各氏共著の記事に記載されている。同じく上記参照誌に、E. Hokenek,R. K. Montoya両氏著の"Leading-zero anticipator(LZA) in the IBM RISC System/6000 floating-point execution unit(RISC System/6000浮動小数点演算実行装置における上位０ビット予測器)"という記事が掲載されている。

【０００５】

【発明が解決しようとする課題】

最上位有効ビット位置予測器からミステリを取り払い、複雑でなく実施と製作が簡単なＬＢＡを実現する手段が必要とされている。

【０００６】

【課題を解決するための手段】

本発明は、ＰＫＧ符号方式を採用する浮動小数点加算器における使用のため、２つのｎビット・オリジナル・オペランドの２の補数２進和の最上位有効ビット位置を予測する改良された方法を提供する。この方法は、

(1)上記２つのｎビット・オリジナル・オペランドを２つの(ｎ＋１)ビット形式に非正規化するステップ、

(2)上記２つの(ｎ＋１)ビット形式の各ｉ番目位置毎に、第１の値が２つの(ｎ＋１)ビット形式のそれぞれｉ番目の２つのビットの加算がゼロであることを示し、第２の値がその加算が１であることを示し、第３の値がその加算が２であることを示す値を持つ３種類の記号の１つをｉ番目記号として割り当てることによって、(n＋１)個の記号シーケンスを生成するステップ、

(3)上記(ｎ＋１)個の記号シーケンスの最左端から開始して、３つの連続する記号を１つのトリプルと定義し、隣接する２つのトリプルが常に共通の２つの隣接記号を持つように(ｎ＋１)個のトリプルを形成するステップ、

(4)上記トリプルの各々を検査して、該トリプルに対応する上記非正規化オペランドのビット位置の部分的加算セグメントが２つの最上位有効ビットを持つことを示す第１の値またはそれ以外を示す第２の値を持つ暫定ビットを、少なくとも上記トリプルの数に等しい数の暫定ビットから構成される２進暫定ワードの対応する暫定ビット位置に作成するステップ、

(5)上記２進暫定ワードの検査によって、上記第１のビット値を持つ最上位有効ビット位置を決定し、決定したビット位置を上記２進暫定ワードの最左端からのビット位置数Ｊで表現するステップ、

(6)上記２つの(ｎ＋１)ビット形式値を加算して未調整和を作成するステップ、(7)上記未調整和を上記Ｊに等しい量だけ正規化するステップ、および、

(8)Ｊが１ビット位置小さすぎることを判断し、Ｊが１小さい場合、Ｊに対応するシフトとは別に更に追加シフトを行うステップ、

を含む。

【０００７】

最下位Ｋの仮定が使われ暫定ビットがセットされない場合、無調整演算和はゼロである。ゼロのケースは指数に関する強制ゼロを必要とするが、正規化は必要とされない。上記ステップ(5)において、残りのすべてのビット位置について暫定ビットを作成するように、ＭＳＢについての規則を若干変更することによって、規則を簡素化することが可能である。その簡素化は、中央の記号がＰであるいかなるトリプルもゼロの暫定ビットを作成することができるというものである。

【０００８】

【発明の実施の形態】

図１に示される浮動小数点加算回路１は、マイクロプロセッサのような集積回路内での実

装に適するように改良された最上位有効ビット予測器(ＬＢＡ)６を装備している。図１の回路は例示の目的のものである点は理解されるであろう。例えば、(オーバーフローを回避するため)既に(ｎ＋１)ビット表示形式にシフトされたｎビット演算を処理するものとして示されている。本発明の好ましい実施形態においてはｎ＝６４であるが、ｎは、それより大きいビットでも小さいビットでも可能である。しかし、そのような特定のｎビットおよび(ｎ＋１)ビット・オペランドは２進数の２の補数形式である。本発明の実施例は、ＰＫＧ符号化形式を使用する加算器を用いる。

【０００９】

本発明のＬＢＡがＰＫＧ符号化方式を活用するとはいえ、ＬＢＡによって必要とされるＰＫＧ符号化はＬＢＡの一部であり、加算器は未符号化２進加数に直接応答することができる点は以下の記述によって明白となるであろう。

【００１０】

参考のため、２の補数の例として、５ビットで表した場合の２の補数の一部を次の表１に示す。

【００１１】

【表１】

0=00000	8=01000	-1=11111	-9=10111
1=00001	9=01001	-2=11110	-10=10110
2=00010	10=01010	-3=11101	-11=10101
3=00011	11=01011	-4=11100	-12=10100
4=00100	12=01100	-5=11011	-13=10011
5=00101	13=01101	-6=11010	-14=10010
6=00110	14=01110	-7=11001	-15=10001
7=00111	15=01111	-8=11000	-16=10000

【００１２】

図１のＡおよびＢは、(ｎ＋１)ビットの２の補数形式の２進数値であり、加算されるべき２つの浮動小数点の有効桁部分、すなわち一般的に仮数と呼ばれるものである。ＡおよびＢは関連する指数部分を持つが本図では示されていない(仮数と指数が組み合わされて完全な浮動小数点を形成する)。２つの指数は既知の方法で結合され、和に関する試行指数２２が作成される。試行指数２２については、図１に関する説明の最後の部分で触れる。

【００１３】

図１の処理の前段階では、２つの(ｎ＋１)ビット仮数は各々ｎビットで表現され、それらの指数はそれぞれ異なっている。浮動小数点加算のためのセットアップの一部として、２つの浮動小数点オペランドは、それらのそれぞれのビット位置が同じ２の累乗数に対応するように、調整される。これによって、オペランドの１つが非正規化されることが必要となることがある。すなわち、加数がアンダーフローを起こすようにその仮数がシフトされ、その指数がそれに応じて調整される。しかし、２つのオペランドが絶対値においてほぼ等しければ、桁あわせのための非正規化は必要ではないが、演算結果がオーバーフローを起こす可能性を生じる。オーバーフローは望ましくなく可能な限り避けられるべきものである。

【００１４】

２の補数の演算のオーバーフローは重大な問題であり、２つの最上位ビットの繰り上げキャリの排他的ＯＲ(ＸＯＲ)によって以前は検出されていた。そのような検出は例えばＯＶのようなフラグ・ビットをセットした。最上位有効桁のビットから出るキャリの値を使用して例えばＥＸのような１ビット拡張レジスタがセットされた。オーバーフローの本当の問題は、アンダーフローと対照的に、その調整が、ＥＸをＭＳＢへシフトさせる一方そのＭＳＢおよびすべての他のビットを右に(下位ビット方向へ)シフトさせることによって行われる点である。アンダーフローの調整方法は、これとは反対に左方向へシフトさせる。この２つのケースは、また、異なる方向の指数調整を必要とする。従って、オーバーフローの好ましくない点は、それが仮数をシフトさせ指数を両方向で調整する必要性を生むこ

とである。

【００１５】

オーバーフロー回避の戦略は、オーバーフローを防止するために十分な非正規化が桁あわせの間に行われない場合、何らか別の措置を講じることを保証するというものである。その何か別の措置とは、ｎビット・オペランドを(ｎ＋１)ビット形式で表すものである。これはその数値を変えることなく、オーバーフローが起こる前により多数のビットを維持することができるメカニズムでそれを表現するにすぎない。これは、例えば、.０７を０.０７として書き直すことに等しい。これは、以下に記述するＰＫＧ表示のビット単位形式に適している。このオーバーフロー回避戦略の要点は、両方の調整を実行しオーバーフローの発生しないことを保証するために必要とされるものとしてオペランドが(ｎ＋１)ビットで書き直され非正規化されるという点である。和の計算後結果が正規化されｎビットに書き直される。オーバーフローがないことが保証される一方、それ以上のビット幅を必要とするオーバーフローがない限り、より広い幅の表示形式でアンダフローを持つという意味において、正規化メカニズムが一層活用される。このような付加的正規化の必要性は、問題でなく、オーバーフロー回避を行わない場合でさえ頻繁に必要となる。対応する指数調整についても同様の点があてはまる。すなわち、オーバーフローを避けることができるならば、指数を両方向に調整する必要性がある。オーバーフローから逃れ両方向で調整するコストは、ｎ＋１ビットの表示形式への事前の非正規化を考慮するに価する。

【００１６】

２つの(ｎ＋１)ビット仮数入力Ａ(図１の符号２)およびＢ(同じく３)がＰＫＧ生成回路４に適用される。ＰＫＧ生成回路４は、ＡiおよびＢiの各ペアからＰi、ＫiまたはＧiの１つを生成するゲーティングを含む既知の回路である。Ｐは"propagate"すなわち伝播を意味し、Ｋは"kill"すなわち無視を意味し、Ｇは"generate"すなわち生成を意味し、それぞれｉ番目のビット位置に関するキャリ・ビットを指す。従って、２つの（ｎ＋１)ビット入力をＰＫＧ生成回路に適用すると、その結果の出力は、(ｎ＋１)ビットＰＫＧワード５となる。これは、(ｎ＋１)個のＰＫＧトリプルとも呼ばれる。各ビット位置毎に、Ｐ、ＫおよびＧは次の定義に従って生成される(ＩＦＦは"の場合に限り"の意味)。

Ki = 1 IFF Ai + Bi = 0 (～Ai AND ～Bi)
Pi = 1 IFF Ai + Bi = 1 (Ai XOR Bi)
Gi = 1 IFF Ai + Bi = 2 (Ai AND Bi)

【００１７】

本発明の好ましいハードウエア実施形態において、Ki、PiおよびGiの各々は、(例えば２ビットのような)信号ペアで符号化するのではなく、(例えば「３つのうちの１つ」という表現のような）対応する３つの相互排他的信号によって個々のビットとして表現される。原理的にはいずれの方法も同様に動作するが、個別表示の方が処理性能がよい。

【００１８】

(ｎ＋１)ビットＰＫＧワード５は、最上位有効ビット予測器(ＬＢＡ)６および加算器キャリ連鎖回路７という２つの場所に送られる。ＬＢＡ６については後回しにして、先ず加算器キャリ連鎖７について記述する。加算器キャリ連鎖回路７は、(ｎ＋１)ビットP_WORD(Ｐワード)８および(ｎ＋１)ビットCARRY_WORD(キャリ・ワード)９を生成する。(ｎ＋１)ビットP_WORD８は、単に、ＰＫＧワード５によって加算器キャリ連鎖回路７に適用されるすべてのＰiである。CARRY_WORD(キャリ・ワード)９は既知の方法で作成される。例えば、Ciをキャリ・ビットＣ0＝G0としてＣi＝Gi ＯＲ (Pi ＡＮＤ Ci-1)という規則のビット単位適用によってキャリ・ワード９が作成される。

【００１９】

P_WORD８およびCARRY_WORD９はＸＯＲ加算器１０に入力され、RAW_SUM(未調整和)１１が出力される。RAW_SUM１１は、正規化されてない生(すなわち未調整)の仮数であり、左へ１回シフトされたCARRY_WORDとP_WORDのビット単位ＸＯＲ演算によって得られる。この演算は、最下位有効桁におけるゼロのキャリ繰り入れおよび最上位有効桁における(あるとした場合の)キャリ繰り越しを伴なう。

【００２０】
RAW_SUM１１はシフト器１２へ入力される。軸シフト器である場合もあるシフト器１２は、アンダーフロー条件を起こさないようにするため数値を正規化する。例えば(１０進数で云えば)、.００１２３　Ｅ９は、１.２３　Ｅ６にされなければならない(負数について２の補数を使用する本発明の実施形態においては、正規化の後、最も左の２つのビットは正数の場合"０１"であり負数の場合"１０"であり、最上位有効桁から２番目と３番目の間に小数点がある)。シフト器１２は、RAW_SUM１１をシフトさせる量を示す次の２つの入力を使用する。すなわち、(1)シフト量を示す２進数値SHIFT_#１３および(2)SHIFT_AGAINと名づけられたシフト器１２への入力として使用されるEXTRA_SHIFT１４である。この信号の出力源についてはＬＢＡ６に関する記述の際詳述する。

【００２１】
RAW_SUM１１はまたオプションの比較器４５にも入力されることもある。比較器４５は、備えられている場合、RAW_SUM１１に加えてすべて０という比較値４６を比較入力として受け取る。比較器４５の目的は、未調整和がゼロであるかどうかを判断し、そうであれば信号２４"を生成することである。

【００２２】
引き続き、SHIFT_#１３が入力Ｂとして、試行指数２２がＢ入力として加算器２１へ適用される。反転されたEXTRA_SHIFT信号１４が繰り入れキャリとして加算器２１に適用される。加算器２１は、Ａ－(Ｂ＋EXTRA_SHIFT)を実行する。これは、全体の演算解の計算された指数値であり、仮数がゼロでないという前提で、加算器２１の指数出力部２７に出力される。浮動小数点加算に関してよく知られているように、仮数ゼロは、指数を無視するか指数を特定の値にセットすることを必要とするが、本実施形態においては便宜上かつ慣例に従ってゼロにする。仮数がゼロであることの検出は、ＬＢＡ６において実施され、ゼロの場合は、別の信号SUM_IS_ZERO２４のソースの役目を果たす信号２４'によって仮数ゼロが標示される。信号２４'が生成されるプロセスは後述する。ＬＢＡ６が必ずしもSUM_IS_ZERO２４のソースである必要はなく、比較器４５によって生成される信号２４"で代替することもできる。

【００２３】
計算された指数２３および事前に符号化されている値ゼロ("ZERO"２６)が２つのデータ入力としてＭＵＸ(マルチプレクサ)２５に適用される。ＭＵＸはSUM_IS_ZERO２４によって制御され、計算された指数２３または"ZERO"２６のいずれか適切な方が加算器２３に関する指数出力２７として出力される。調整済み指数２７および正規化仮数１５が合わされてオリジナル入力ＡおよびＢの最終加算結果となる。

【００２４】
上述したように、加算メカニズムは必ずしもＰＫＧ符号化方式を使用するタイプのものである必要はなく、入力ＡおよびＢに直接接続する完全加算器からなるものでもよい。この点は以下の記述からも明らかとなるであろう。

【００２５】
以下、最上位有効ビット予測器(ＬＢＡ)の記述を行う。ＬＢＡは、その内部動作と関係なく、上位ビット１検出器か上位ビット０検出器かのいずれかである。特定の最上位有効ビット予測器６に対しての説明は膨大となろう。本明細書の記述は、その演算の正確性を証明する目的ではなく、いくつかの例を用いながら、ＬＢＡの機能性の基となる理念を説明する解探索プロセスの概要を説明するにとどめる。いくつかの演算規則を説明し、概念全体は表を用いて示す。

【００２６】
既述のように、ＬＢＡの目標は、演算結果を検査してシフトおよび指数調整によって正規化を行うため、主加算メカニズムによって実際の未調整解が生成されるのを待つ必要性を回避することである。その代わりに、ＬＢＡ６は「正規化量」(SHIFT_#１３)という一般的に信頼性の高い近似値を迅速に形成する。未調整和が生成され次第若干量の検査でそれを正規化することができる。これによって時間が節約される。

【００２７】
しかしながら、問題がないわけではなく、「一般的に信頼性の高い近似値」という概念を使用しなければならない。(ＬＢＡの目的にとって自滅的ではあるが)主加算器自体と等しい処理量を行わない限り、近似値は、ある場合は正確であり、ある場合はカウント１外れる。そうなる理由を理解し、いくつかの適切な選択を行うことによって、エラーが発生する場合エラーが常に同じ方向にあるように(例えば１カウント低いように)構成し、エラーが発生する場合を検出しそれを容易に調整する修正対策(すなわちEXTRA_SHIFT１４信号の発信)を工夫することができる。この修正対策の負荷は、最上位有効ビット予測器を使用しない加算－検査－正規化というアプローチに比較して非常に少ない。
【００２８】
正規化量の近似値が得られるプロセスを以下記述する。目標は、単に出発点として、未調整和の近似値を生成し次に検査を行うことと仮定する。検査は、実際の未調整和が手元にあるとすれば実行されるであろうものと全く同じであるので、近似値未調整和から得られる時間的性能利得は、未調整和に対して必要とされる時間と比較して小さくなければならない。この目的を達成するために必要な情報が加数ＡおよびＢのＰＫＧ表示形式に含まれている。ＰＫＧワード５から別のAPPROX_SUM(近似和)ワードを生成する試みをＰＫＧワード内の複数の記号から１時点に１記号ずつ取り出すように実施すると仮定すると、この試みの命運は直ちに定まる。すなわち、Ｐ、ＫまたはＧが、ｉ番目のビット位置のための記号であるか否かに関係なく、その位置に関する繰り入れキャリの存在または不在が和のｉ番目のビットの値を変えるので、ただ１つの記号から十分な情報を得ることはできない。
【００２９】
次の試みは、ＰＫＧワード５の隣接する記号ペアを検査することでる。あらゆるケースではないが、いくつかの結果を以下に示す。ビットｉおよび(ｉ－１)に関するストリングＫＫを考察する。(本明細書ではある数のビットを一般に行われているように左から右へ書き、最右端をビット・ゼロとする)。この場合、両方のビット位置の和はゼロであり、(ｉ－１)位置への繰り入れキャリはそこにおかれたままであるので、ｉ番目の位置はゼロであるということができる。あいにく、ビットｉおよび(ｉ－１)がストリングＰＰであるようないくらか曖昧なケースがある。この場合(ｉ－１)番目の位置の繰り入れキャリに依存するということを除いて、ｉ番目の位置については何も云うことができない。
【００３０】
負数の場合１という最上位ビット・ストリングおよび正数の場合０という最上位ビット・ストリングに関してAPPROX_SUMを検査することも以下のアプローチに含まれる。しかし、正または負は、最上位ビットによって示されるが、この最上位ビットはそこにキャリされるものによって影響される。和の符号に関する不確実性は快適でない。また、最上位１ストリングに関する検査を行うハードウエア・セットと最上位０に関する検査を行う別の第２のハードウエア・セットを備える必要性を回避することが可能であるか明確でない。
【００３１】
正負に関する不確実性に対する解決策は、APPROX_SUMを一緒に実行し、未調整和におけるビット反転の近似値を出力するようにＰＫＧワード検査メカニズムを調整することである。本明細書において、この近似を「マップ」と呼び、「近似暫定マップ」をAPPROX_X_MAPと名付ける。マップにおける連続的０ストリングは、実際の未調整和の中の対応するビットがすべて同じ(１または０)であることを意味する。マップのある１つのビット位置における１は、その位置の左にある未調整和のビットが異なっている、換言すれば現在の位置が反転を表すことを意味する。そのようなマップを持つことができれば、正であるか負であるかによって煩わされることなく、検査によってAPPROX_X_MAPの最も左の１を検出することができる。このマップについて近似的であることは、反転が実際に起こるところから１ビット位置左にある実際の未調整和の位置をマップの中の１が示す可能性があるとうことである。このため、この方法が上述のように問題がないわけではないということになる。
【００３２】

次に、ＰＫＧワード５の３つの記号を一度に検査する方法を考察する。この方法は、ＰＫワード５の左側(最上位有効部)から始め、ＰＫＧ記号の最上位有効トリプル(すなわち３連続記号)を取り出し、次に続くトリプルを取り出しながら一度に１記号を処理する作業を繰り返す。このように、１つのトリプルから次のトリプルへ２つの記号が重複する。１つのトリプルが検査される毎に、APPROX_X_MAPにおける対応するビット位置に関する値が生成される。

【００３３】

トリプルの３つのどの位置もＰ、ＫまたはＧの３つのいずれかが占めることができるので、考察すべきトリプルを形成する組合せは２７ある。表１に示されるように、トリプルのどれも、未調整和１１の対応する部分的セグメントに関する暫定ビット値(bit value transitions)を示すが、全体ワードのトリプルの位置が右に少くとも１記号あるようなケースに関しては(大部分このケースである)、該トリプルに対応する和のセグメントの右側部分へのキャリ繰り入れが存在することもしないこともある点は注意する必要がある。最上位トリプルがAPPROX_X_SUMの最上位ビットを決定し、下位の方向に進むにつれそれに対応したビットを決定する。オーバーフロー回避のため(ｎ+1)ビットへシフトされたオリジナルのｎビット・オペランドに対して、(ｎ－２)個のトリプルおよびAPPROX_X_SUMにおける対応する(ｎ－２)個のビット位置が存在する。このような点が、(ｎ＋１)ビット未調整和に関する最上位有効ビット検出にとっていかに有効な情報であるかは、図２を参照して後述する。

【００３４】

次の表２に関連して、文字？を使用する。？は「不問」または「ワイルド・カード」を意味し、その位置がビット１であろが０であろうが、あるいは、Ｐ、Ｋ、Ｇのいずれの記号でも関係のないことを示す。

【００３５】

【表２】

トリプル	結果(RESULT)	キャリあり の和	キャリなし の和	理由
KKK	0	000	001	#1
KKP	1	001	010	#2
KKG	1	010	011	#4
KPK	1/0(MSB/～MSB)	010	011	#3
KPP	1/0(MSB/～MSB)	011	100	#3
KPG	1/0(MSB/～MSB)	100	101	#3
KGK	1	100	101	#4
KGP	1	101	110	#2
KGG	0	110	111	#1
PKK	1	100	101	#4
PKP	1	101	110	#2
PKG	0	111	000	#1
PPK	0/1(～LSB/LSB)	110	111	#1,#5
ppp	0	111	000	#1
PPG	0	000	001	#1
PGK	0	000	001	#1
PGP	1	001	010	#2
PGG	1	010	011	#4
GKK	0	000	001	#1
GKP	1	001	010	#2
GKG	1	010	011	#4
GPK	1/0(MSB/～MSB)	010	011	#3
GPP	1/0(MSB/～MSB)	011	100	#3
GPG	1/0(MSB/～MSB)	100	101	#3
GGK	1	100	101	#4
GGP	1	101	110	#2
GGG	0	110	111	#1

#1：各部分的和の左２ビットが反転を示さないので、０であるにちがいない。
#2：？ＫＰと？ＧＰには、各々が左２ビットが反転し、残りが反転しない部分的和を常に持つので、１を割り当てる。この結果は、正しいか、あるいは、有効ビットより１桁プラスであるかもしれない。
#3：ＫＰ？およびＧＰ？は、１を生成する？ＫＰおよび？ＧＰの右側に起こるので、不問のケースであり、これらと中央にＰがあるトリプルには０を割り当てる。
#4：ＫＫＧ、ＧＧＫ、ＧＫＧ、ＫＧＫ、ＰＫＫおよびＰＧＧは、中央に明確な反転がある。
#5：マイナス１（すべて１）の仮数とゼロ（すべて０）の仮数を区別する。

【００３６】
上記表２は、２７の可能なＰＫＧトリプルおよびRESULT(結果)に関するそれらの対応する値のリストである。RESULTは、歳差連動のインスタンスに対応するAPPROX_X_MAPにおけるビット位置の値である。すなわち、対象がトリプルを考察する５番目のインスタンスであれば、RESULTはAPPROX_X_MAPビットの(左から数えて)５番目に当てはまる。ここでの記述は、表２の結果(RESULT)欄で" 1/0 (MSB/～MSB)として表示されているトリプルＫＰＫ、ＫＰＰ、ＫＰＧ、ＧＰＫ、ＧＰＰおよびＧＰＧが１として解読される規則を含むＭＳＢから始める。各トリプルおよびその対応するRESULT値が２つの異なる和セグメントを伴うという点に注意する必要がある。これらの和セグメントは、歳差連動の回数に応じて、ＰＫＧトリプルの位置によって表されるビット位置に対応する未調整和１１のビット位置に位置する。これらの和は、発生し得る(唯一)可能な和セグメントであり、セグメントの右側にキャリ繰り入れがあるか否かに関連して、記号Ｐ、ＫおよびＧを最初の場所に割り当てるために使用される規則を考慮することによって得られる。ここで、オーバーフローを回避するため(ｎ＋１)ビット未調整和１１を使用していること、また、APPROX_X_MAPのＭＳＢに関するRESULTが１であるということはアンダフローが発生しなかったことを意味するという点に注意する必要がある。実際には、次の２つのケースの１つが考えられる。すなわち、(1)（ｎ＋１)ビット加算にアンダーフローは実際になかった(ｎビット加算にオーバーフローがあったかもしれないことを示唆する)、および(2)アンダーフローがないという標示は偽りであって、１というRESULTは０で次のインスタンスのRESULTが１であるべきであった、という２つのケースである。この第２のケースをＬＢＡエラーと呼ぶ。アンダーフローがある他のすべてのケースでは、RESULTは０でなければならない。

【００３７】
アンダーフローもオーバーフローもない２の補数和は正数については０１？で負数については１０？で始まる。和が(正の場合)００？または(負の場合)１１？として始まるケースはアンダーフローを表す。表２の説明を開始する最も簡単な箇所は、０というRESULTがどのように調整されるかに関するものである。これらは、アンダーフローの発生に関して不確実性のないケースである。

【００３８】
これらのケースは、表２において異なるトリプルを伴なう種々の加算セグメントを調べることによって見つけられる。加算セグメントについて常に０００または１１１だけを生成するトリプルは、RESULTの値について０を復号しなければならない。このようなトリプルはＰＫＧおよびＰＰＰの２つである。加算セグメントについて常に０００または００１だけを生成するトリプルもまたRESULTの値について０を復号しなければならない。そのようなトリプルは、ＫＫＫ、ＰＰＧ、ＰＧＫおよびＧＫＫの４つである。最後に、加算セグメントについて常に１１１または１１０だけを生成するトリプルも同様にRESULTの値につい

て０を復号しなければならない。そのようなトリプルはＫＧＧおよびＧＧＧの２つである。これら９個のトリプルだけが、キャリ繰り入れがあろうとなかろうと、アンダーフローが確実なトリプルである。これらのトリプルのRESULTは、絶対に０でなければならない。

【００３９】

同様に、絶対にかつ明確にRESULTが１を持つトリプルを見い出すことができる。このためには、唯一の可能な加算セグメントが０１？または１０？の形式をもつトリプルを捜す必要がある。これらは、ＫＫＧ、ＫＰＫ、ＫＰＰ、ＫＰＧ、ＫＧＫ、ＰＫＫ、ＰＧＧ、ＧＫＧ、ＧＰＫ、ＧＰＰ、ＧＰＧおよびＧＧＫという１２個のトリプルである。これらのケースの各々は、未調整和１１がアンダーフローを示さないインスタンスを表し、従って、また、ＬＢＡエラーのインスタンスではない。

【００４０】

これら１２個のトリプルのケースが、部分的加算セグメントの対応する各ペアが１というRESULTを生成すべき１つのセグメントおよび０というRESULTを生成すべきもう１つのセグメントを持つすべてのケースである点に注意する必要がある。部分的加算セグメントのそれらビットへの実際のキャリ繰り入れに関する情報を持たないが、１というRESULTを生み出す必要性を無視することはできない。解決策は、たとえそれが０でなければならなかったとしても、そのようなインスタンスの各々は実際にＬＢＡエラーであるので、１というRESULTを生み出すことが安全である点に注目することである。すなわち、０でなければならなかったのに１というRESULTを形成したとしても、「観念上の」次のRESULT値がとにかく１でなければならなかったことは確実である(実際の次のRESULT値については何も云えないが、１というRESULTが１ビットすぐ後に発生する時補正することができるということだけは云える)。

【００４１】

従って、キャリ繰り入れがなければトリプルＫＫＰ、ＰＧＰおよびＧＫＰは０というRESULTを生成すべきであり、一方キャリ繰り入れケースを(RESULTが１であるべき)アンダーフローなしの正数ケースに合致させることが表２から観察される。同様に、トリプルＫＧＰ、ＰＫＰおよびＧＧＰは、キャリ繰り入れのある負のアンダーフローおよびキャリ繰り入れなしのアンダーフローなしを表すことが観察される。しかし、これらのインスタンスの各々もまたＬＢＡエラーのインスタンスであるので、状況は救われる。例としてＫＫＰを考察する。部分的加算セグメントが実際にキャリ繰り入れに関する結果である００１であったと仮定する。００１は一度左にシフトすることによって正規化されなければならないので、エラーである１というRESULT(アンダーフローなしおよび正規化なし)が生成される。今検討していることは、未調整ビットの最も左の３ビットが００１であるというケースである。従って、歳差連動における次のトリプルは、０１？という形式の加算セグメントを標示することが予想される。これは正の２の補数値の有効桁の始まりである。ＬＢＡエラーが発生したことを検出できるように構成することができると仮定すると、それを直すには、更に１ビット位置正規化を行うことだけでよい。これは、本ケースでは、APPROX_X_MAPのＭＳＢにおける１というRESULTによる１シフトの全体正規化を意味する。一方、ＫＫＰに関する部分的加算セグメントが０１０であったとすれば、APPROX_X_MAPのＭＳＢにおける１というRESULTはエラーでない。この場合ＬＢＡエラー・メカニズムが起動されないことを確認することだけが必要とされる。残りのトリプルＰＧＰ、ＧＫＰ、ＫＧＰ、ＰＫＰおよびＧＧＰは、対応する同様の例によって説明できる。これで、APPROX_XSUMのＭＳＢをセットする説明を終わる。

【００４２】

次に、APPROX_X_SUMの残りのビットを決定する作業の説明に移る。直前のトリプルがＭＳＢに関するものであったと仮定すると、(ＬＢＡエラーの場合であろうとなかろうと)、ＭＳＢに対する分析によって既に１というRESULTまたは０というRESULTのいずれかが出力されている。RESULT０はアンダーフローが発生したことを保証する。これは、RAW_SUM１１の最も左に連続する少なくとも２つの０または１があることを意味する。

【００４３】

この場合、トリックではあるが、最も左のビットを破棄して１トリプルだけ進めあたかもＭＳＢに関してもう一度すべてをチェックすることを考えてみる。これは、いずれにして実際に有効桁をシフトさせることによって正規化を行う機会が来る時RAW_SUM１１に対して実行することに対応していることであるので、このように最も左の０または１のストリングを一度に１ビット破棄することができる点は明白である。この点についての懸念は、表２の観察によって解消される。１１？または００？を含む部分的加算セグメントはいずれも０というRESULTを生成する。仮に切り落とされたRAW_SUMがビット反転の前になお５個の１か５個の０を持っているとすれば、１というRESULTが来る前に少なくとも４つの「偽のＭＳＢ破棄」が存在すること(APPROX_X_MAPの対応するビット位置の少くとも４つが０であること)が保証される。これは、有効ビット１の前に例えば５つの０があれば次の４回の歳差連動が０００以外の部分的加算セグメントを持つことはないので、うまくはたらく。なぜなら、０００を持つとすれば、５個の０の中に１つの１が含まれていることになるがそういうことは仮定上ないからである。これはRESULTの次の４つの値が０であることを意味する。００１または１１０の形式の部分的加算セグメントが発生するのは早くても「破棄」ストリングの最後であるとしても、そのような形式の部分的加算セグメントが存在するので、５番目の値は、ＬＢＡエラーであるかもしれないしそうでないかもしれない。

【００４４】

重要ないくつかのポイントを以下要約する。正規化された正数は、最上位ビット０１で始まり、正規化された負数は最上位ビット１０で始まる。正規化を必要とする正数は、左端にいくつかの０のストリングを持ち、一方、正規化を必要とする負数は左端にいくつかの０のストリングを持つ。数は、(ゼロを含む)正数かあるいは負数のいずれかであるので、左端にすべて１またはすべて０のストリングがあり、左端の符号ビットのストリングの検査を開始すれば、トリプルを右へ順次処理する過程で、そのトリプルの対応する部分的加算セグメントがその中央または左の位置に反対の値のビットを持つトリプルに突然出会うことを懸念する必要はない。それは簡単に発生しない。ビット値の変化は、対応する部分加算セグメントのかなり右側で最初に起きるであろう。

【００４５】

また、この点で物理的シフトまたはビット破棄を行う必要はない点は明きらかである。複数トリプルにおける「歳差連動」で十分である。更に、トリプルを１回に１つずつ取り扱わねばならないことはなく、また特別な順序で処理する必要もない。なぜならば、トリプルは一旦取り出されれば固定的であり、表２に基づくそれらの解釈は、トリプルがＭＳＢに対するものか否かに関することのみに依存するからである。このように、すべてのトリプルは、独立していて、APPROX_X_MAPに対するすべてのビットを同一クロック・サイクルで生成するためにそれらを並列的に復号することができる。このように、「歳差連動」の概念は、単に理解のための便宜的装置にすぎない。その単一クロック・サイクルの後、主ＬＢＡが残す信号は、APPROX_X_MAP中の最も左の１のビットの位置およびＬＢＡエラー検出の識別である。

【００４６】

これらの事項に記述を進める前に、表２に関して残っている問題を取り上げる。その第１は、ＭＳＢについて処理しているか否かに基づくRESULTの値の相違の問題である。ＭＳＢの場合RESULTが１である理由は既に記述した。非ＭＳＢトリプルの場合それが０に変わる理由を明確にする必要がある。トリプルがＫＰ？およびＧＰ？で標示される６個のトリプルである場合にこの点が明らかになる。一旦それらがＭＳＢトリプルでないことがわかると、それらは、必ず？ＫＰおよび？ＧＰの右で発生することがわかる。しかし、それら６つのトリプルは常に１というRESULTを持つ。従って、(APPROX_X_MAPにおいて)そのような１の右側にあるRESULTの値は重要でなく、「不問」の値である。それらのトリプルについてRESULTの値に０をセットする場合、(非ＭＳＢ)トリプルのＰという中央記号がＲＥＳＵＴ値が０であるべきことを示すに足るものであることが表２によって明らかである。これは、RESULTの値を生成するためにトリプルを復号する論理における望ましい簡略化である

。

【００４７】

別の問題は、APPROX_X_MAPのＬＡＢおよび発生可能な例外ケースにかかわるものである。例えば、APPROX_X_MAPはすべて０を受け取ることができるであろうか？受け取ることができるとすればそれは何を意味するのであろうか？　また、１つのトリプル毎にAPPROX_X_MAPの１ビットが対応しなければならないと述べた。しかし、加数が(オーバーフロー回避の後)各々ｎ＋１ビットであるならば、ＰＫＧワード５に(ｎ＋１)個のＰＫＧ記号があり、それは(ｎ－1)個のトリプルしかないことを意味する。このことは、トリプルに関して処理するＬＢＡが右方向へ十分な長さを持っていないことを意味すのではないか？　この問いは、他の関連した問題とともに、図２を参照することによって回答される。

【００４８】

図２は、２の補数加算のビット処理における５つの異なる段階を示している。図２の最上部には、２つのオリジナルのｎビット・オペランド、Ａ'２８およびＢ'２９が示されている。オーバーフロー回避の目的のため、これらは(ｎ＋１)ビット・オペランドＡ(３０，２)およびＢ(３１，３)に算術的に右にシフトされている(算術的シフトは符号ビットを維持する)。Ａ'およびＢ'から(ｎ＋１)個のＰＫＧ記号(３２、５)が作成される。種々のダイヤモンド３５は、２進オペランドのペアからのＰＫＧ記号のビット毎の生成に関する規則を示す。ＰＫＧ記号(３２、５)は、種々の括弧３６によって示される(ｎ－１)個のトリプルに分類される。表２において記述された関係を実施するゲーティング機構によって、各トリプルは、種々の$RESULT_i$を生成する。種々の$RESULT_i$は、APPROX_X_MAP３３におけるそれぞれの(ｎ－１)ビット位置に対応するビット値をセットする。そのようなAPPROX_X_MAPの(ｎ－１)ビットは、図に示された方法で、最終的RAW_SUM３４の(ｎ＋１)ビットに対応する。

【００４９】

最後に、図２の特定の重要なケースを説明する前に、ＬＢＡの機能をよりよく理解するため図２が主に諸ビット位置間の関係を示すように描かれている点を明記する。図２は、(推論できる箇所はあるが)レジスタ間のハードウエア接続を必ずしも表してはいない。特に、図は、APPROX_X_MAP３３がRAW_SUM３４に対してビットを供給することを示唆していない。ビット間の対応関係を記述しているに過ぎない。

【００５０】

図２の説明に戻り、既に述べたようにAPPROX_X_MAPがすべて０を含むと想定すると、これは、未調整和１１がビット反転の前に最上位ビット・ストリングを含まないことを意味する。未調整和１１はすべて１かすべて０である。RAW_SUM３４のビットより２つ少ないトリプル(またはAPPROX_X_MAP３３のビット)があるけれども、上記の点は確かであろう。あいにく、ＬＳＢ(最下位有効ビット)に関する情報は十分でない。RAW_SUM１１が、ゼロ(すべて０)か、マイナス１(すべてｌ)かあるいはマイナス２(ＬＳＢが０であることを除いてすべて０)のいずれかであることは最低限確実である。状況を明確にするため、先ず、APPROX_X_MAPの最上位の１ビット値が０であることがRAW_SUM３４最上位２ビットについて意味する点を知る必要がある。それは和が少なくとも１シフトだけ正規化されなければならず、従って和上位ビットは１１か００でなければならないことを意味する。和が１１で始まりビット値反転がなければ、それは１１１１...でなければならない。すべて０についても同様である。第２に、APPROX_X_MAPのＬＳＢにおける０というビット値はRAW_SUMのＬＳＢを決定するために十分でないことが観察される。それは、その左のビットが０であれば０であるが、その左のビットが１であればＬＳＢが何かはわからない。この理由を理解するため図１を再び参照すると、キャリ繰り入れの可能性がなく、(仮定によってその左に１がないので、すなわち上述のようにＫＰ？およびＧＰ？は１を生成するトリプルの右にのみ存在するので)トリプルＫＰ？およびＧＰ？は発生しないという追加情報が得られる。

【００５１】

対象となるＫＫＫ，ＫＧＧ，ＰＰＫ，ＰＰＰ，ＰＰＧ、ＰＧＫ、ＧＫＫおよびＧＧＧとい

う８つのトリプルのうち、ＫＫＫ、ＰＰＧ、ＰＧＫおよびＧＫＫに関する部分的加算セグメントはキャリ繰り入れなしのすべて０である。これは、左のビットが０である時ＬＳＢも０であるというケースである。ＰＰＰはすべてが１であるというケースに関連する。あいにく、ＫＧＧ、ＰＰＫおよびＧＧＧという３つのトリプルの場合は明確でない。それらは、各々、１１０というキャリ繰り入れのない部分的加算セグメントを持つ。従って、左のビットが１であるがＬＳＢが０であるというケース(マイナス２という和を形成するケース)が存在する。このような状態は、マイナス２を報告しなければならない場合なのにマイナス１という未調整仮数をＬＳＢが報告する可能性を生む。そのような誤った報告によって不正確な量の正規化が行われ、許容できない演算エラーが発生する。

【００５２】

しかし、上述の問題点が図２に従う実施形態が全く役に立たないことを意味しているわけではないことは理解されるべきである。上述の明確でない点を取り除き演算エラーが発生する可能性のない別の外部環境を存在させる可能性はある。例えば一層大規模な乗算累算装置の一部としての浮動小数点加算器においてＬＢＡが使用される場合、そのような構成を実施することが可能である。その理由にこだわる必要はないが、上述のエラー・メカニズムはもはや呼び出されることはない点は明らかである。図２に従う実施形態は分析目的に主眼がおかれてはいるが、加算累算装置の部品として実際の製品に組み込まれたものがある。

【００５３】

しかしながら、マイナス１およびマイナス２という未正規化仮数の間のこのような不便なＬＳＢの混乱を簡単に除去し、ＰＫＧ記号のトリプルに関する「歳差連動」の比較的単純な技術を独立型加算器において使用することができることが望ましい。これは、図３において示されるような形態で簡単に実施できる。

【００５４】

図３は、多くの点で図２と同じである。相違する点は、図３が、値が「Ｋ」である最下位ＰＫＧ記号３７の追加を想定している点である。これは、最下位トリプル３８から付加的最下位RESULT0３９を生成することを可能にする。追加のRESULT0を持つことによって、(今や拡張された)APPROX_X_MAPの最下位にもう１つのビット４０を持つことを可能にする。APPROX_X_MAP４１における付加ビット４０は、RAW_SUM３４との正しい対応関係にある。このようにして物事は想定した通りにはたらく。

【００５５】

次に、ＰＫＧ記号の最下位の位置にＫがあると推定することが合理的である理由を考察する。結果として生成されるビット・パターンに関する限り、追加すべきｎビット２進数を取り上げ単に右にゼロを付加することが常であり、左ビット・パターンは変わらないままである。「暫定的に」負の数が維持されないことは真実であるが、２の補数表現は単にビット解釈であり、いずれにしても、２の補数および符号なし２進整数は、同じようにビット毎に追加される。更に、図３において余分なゼロは右に追加されない。APPROX_X_MAP４１にビット４０が追加されればその必要はない。

【００５６】

RAW_SUM３４の(ｎ＋１)ビットを見つける方法も変わらない。従って、一時的に、０という余分な低位ビットを持ちながら非正規化数Ａ３０およびＢ３１を暫定的に構築する点に問題はない。構築するとすれば、そのような０はＫというＰＫＧ記号を生成するであろう。これがＫ３７が作成される経緯である。かくして、「トリプル」の通常の「歳差連動」および「ＭＳＢ削除」分析がRAW_SUM３４のＬＳＢに対して直接はたらく。

【００５７】

最後に、ＰＰＫに関する表２のエントリを考察する。0/1(～LSB/LSB)という表記は、仮定された追加Ｋ３７の使用を仮定する。それ以外は、ＰＰＫに対するRESULTはゼロである。この特別なケースは、RAW_SUM１１がマイナス１(すべて０)の時APPROX_X_MAPがすべて０を含むことを防止するため、表２に含まれる。(ＰＰＫは、１１０というキャリ繰り入れのない部分的加算セグメントを持つ唯一のＫで終わるトリプルであって、０であるRAW_SU

Mの実際のＬＳＢの左に１がある)。１をAPPROX_X_MAPのＬＳＢに強制的に挿入することによって、マイナス１という未調整仮数が正規化を不要とすることが防止される。その代わりに低位方向へ１カウントという正規化量が示される。すなわちそれはＬＢＡエラーである。この点は問題ではなく、通常のように検出され修正される。これを達成するために必要な論理は、RESULT0を作成するゲートと同様であるが、ただし、(ｎ＋１)ＰＫＧ記号３２、５における実際のＬＳＢ記号(すなわち、０の添え字を持っＬＳＢ記号)がＰであれば、出力を１にするという条件を持つ。

【００５８】

次に図１に戻って、ＬＢＡ６の内部構造を考察する。ＰＫＧシーケンス検出器１７は、表２を実施するビット単位ゲーティング機構である。このゲーティング(あるいはおそらく参照用テーブル)が、その入力側でＰＫＧワード５へ接続され、その出力はAPPROX_X_MAP(３３、４１)のビットである。それらビット(種々のRESULTi)が上位１キャリ連鎖回路１９に適用される。キャリ連鎖１９は、APPROX_X_MAPにおける最上位１のビット位置Ｊを見つける。この数Ｊは左からのビット位置のカウントである。Ｊがゼロであれば、RAW_SUM１１は、(その右端のビットを取り除くことによって切り捨てられる必要があるけれども)正規化される必要はない。出力１３のSHIFT_#(シフト量)はＪに関する値のバイナリ符号である。出力２０は、「ｎの１つ」であるＪの値を表す。この場合、図３に示される実施形態の場合「ｎ」はオリジナルのオペランド(２８、２９)にあるものと同じビット数であり、図２の実施形態の場合はそれより１少ない。注意すべき点であるが、ＬＢＡ６が、SUM_IS_ZERO(和ゼロ信号)２４のソースであるべき信号２４'を作成する責任を果たすため、RESULTiのすべてが０でありRAW_SUMのＭＳＢもまた０であることを適切なゲーティング機構を用いて検出することによって、その信号が作成される。代替的方法としては、比較器４５を使用して、ＬＢＡ６からの信号２４'がない場合にSUM_IS_ZERO２４に対するソースの役目を果たす信号２４"を生成する。

【００５９】

ＬＢＡエラー検出器１２の動作が図４に示されている。図４には、図２および図３の場合と同様にAPPROX_X_MAP３３，４１およびRAW_SUM３４のセグメントが示されている。APPROX_X_MAPの内容が、最上位１キャリ連鎖回路１９に入力され、そこで、最左端ビットが識別され、右側のビットがすべて削除される。RAW_SUM３４における隣接ビット位置のペアの各々について、その隣接ペアのビットが排他的ＯＲゲート４２に適用される。図４にそのようなＸＯＲゲートがただ１つ示されているが、RAW_SUM３４に存在する多数の隣接ビット・ペアの数に相当するゲート２４が備えられる点は理解されるであろう。ＸＯＲゲート４２の出力は、関連ＡＮＤゲート４３への１方の入力となり、その他方の入力は、ＸＯＲゲート４２へ送られるRAW_SUMのビット・ペアの最下位ビットに対応する最上位１キャリ連鎖回路１９のビット２０である。同様に、４３のようなＡＮＤゲートの別のものがただ１つゲート４４として図示されているが、RAW_SUM３４における隣接ビット・ペアの各々についてＡＮＤゲート４３と同等のものが備わる。ゲート４３、４４の出力のすべてのＯＲ信号は、ＬＢＡエラーの発生を標示するEXTRA_SHIFT信号１４である。APPROX_X_MAPの中の所与のビット位置が最左端であれば、RAW_SUMにおける対応するビット位置およびその先行位置の値は異なると通常予測される。そのような条件の不在はＬＢＡエラーであり、その場合は、種々のＡＮＤゲート４３、４４のうちの１つの出力をＴＲＵＥにする。

【００６０】

図３において、優先キャリ連鎖回路が、次式の規則に従って、添え字ｎのＭＳＢから添え字０のＬＳＢへキャリＣiを伝播する。

Cn = 0、 Ci-1 = (Ci) OR (APPROX_X_MAPi)、および、

Ji = (APPROX_X_MAPi) AND (～Ci)

【００６１】

図５は、表２のＭＳＢ位置以外のすべてを実施する実際の実施形態の１つにおいて使用されたＣＭＯＳゲーティング・セルのブロック図である。出力は、LEADING_BIT_HIGH(最上位ビット高)およびLEADING_BIT_LOW(最上位ビット低)を意味するＬＨおよびＬＥである。

その相補信号は--Hおよび--Lと表記される。最上位ビット信号(ＬＨおよびＬＬ)はトリプル毎に１回出され、既述した種々のRESULTiに対応する。入力に関して、記号Ｐ、ＫおよびＧが観察される。これらは、ＰＫＧワード５のｉ番目の位置に対応する非相補信号を示す。そのｉ番目の位置は、Ｐ、ＫまたはＧのいずれか１つの記号を含む。形式Ｐｐ,Ｋｐ...の記号ペアはＰi+1、Ｋi+1．．．を表し、ｐは左方向へのプラスを意味する。同様に、形式Ｐｍ,Ｋｍ...の記号ペアはＰi-1、Ｋi-1．．．を表し、ｍは右方向へのマイナスを意味する。

【００６２】

既述の通り、表２における関係は、トリプルがＭＳＢに関するものであれば、６つのトリプルについて異なる。ＭＳＢのケースを実施するゲーティング・セルを制作することは当業者にとって可能であろう。しかし、そのような装置を制作するか否かは、許容されるゲート遅延数、チップ上の使用可能スペースおよび単純な個人的好みを含む因子に基づくことであろう。図６および図７は可能な方法を示す。

【００６３】

図６は、図２および図３と同様であるが、相違する点は、ＰＫＧ記号３２、５の最上位ペア４７とPPROX_X_MAP３３、４１のＭＳＢの接続の部分である。特に、図６の場合、APPROX_X_MAP３３、４１の最上位ビットはRESULTnまたはRESULTn+1のいずれかによってセットされることができる。RESULTn+1は、RESULTn+1 = (KP or GP)という関係に従ってＰＫＧ記号の追加最上位ペア４７によって作成される。表２の観察によれば、それが、ＭＳＢケースにおいて、１というRESULTを生成しなければならないＫＰ(？ＫＰ)またはＧＰ(？ＧＰ)であるので、上記の点は機能する。図７は、RESULTn+1 = (KP or GP)という関係を実施するＣＭＯＳゲーティング・セルのブロック図である。図５に関して使用したものと同じ表記法が図７にも適用されている。

【００６４】

本発明には、例として次のような実施様態が含まれる。

（１）２つのｎビット・オリジナル・オペランドの２の補数２進和の最上位有効ビット位置を予測する方法であって、上記２つのｎビット・オリジナル・オペランドを２つの(ｎ＋１)ビット形式に非正規化するステップと、上記２つの(ｎ＋１)ビット形式の各ｉ番目位置毎に、第１の値が２つの(ｎ＋１)ビット形式のそれぞれｉ番目の２つのビットの加算がゼロであることを示し、第２の値がその加算が１であることを示し、第３の値がその加算が２であることを示す値を持つ３種類の記号の１つをｉ番目記号として割り当てることによって、(n＋１)個の記号シーケンスを生成するステップと、上記(ｎ＋１)個の記号シーケンスの最左端から開始して、３つの連続する記号を１つのトリプルと定義し、隣接する２つのトリプルが常に共通の２つの隣接記号を持つように(ｎ＋１)個のトリプルを形成するステップと、上記トリプルの各々を検査して、該トリプルに対応する上記非正規化オペランドのビット位置の部分的加算セグメントが２つの最上位有効ビットを持つことを示す第１の値またはそれ以外を示す第２の値を持つ暫定ビットを、少なくとも上記トリプルの数に等しい数の暫定ビットから構成される２進暫定ワードの対応する暫定ビット位置に作成するステップと、上記２進暫定ワードの検査によって、上記第１のビット値を持つ最上位有効ビット位置を決定するステップと、上記決定したビット位置を上記２進暫定ワードの最左端からのビット位置数ＪＪで表現するステップと、上記２つの(ｎ＋１)ビット形式値を加算して未調整和を作成するステップと、上記未調整和上記Ｊに等しい量だけ正規化するステップと、を含む最上位有効ビット位置予測方法。

（２）Ｊが１ビット位置小さすぎることを決定するステップと、Ｊが１小さい場合Ｊに対応するシフトとは別に更に追加シフトを行うステップと、を更に含む上記（１）に記載の最上位有効ビット位置予測方法。

（３）上記オリジナルのオペランドが指数を持つ浮動小数点数の仮数であって、試行指数を(Ｊ＋１)桁調整するステップを更に含む上記（２）に記載の最上位有効ビット位置予測方法。

（４）上記２つの(ｎ＋１)ビット形式値の実際の最下位有効ビットよりも１ビット小さい

ビット位置に第１の値の記号が位置していることを示すように、上記トリプル・シーケンスに最下位トリプルを追加するステップを更に含む上記（１）に記載の最上位有効ビット位置予測方法。

【００６５】

【発明の効果】

本発明によって、従来技術に比較して単純化された最上位有効ビット位置予測器が提供され、浮動小数点加算器の製造がより容易となる。

【図面の簡単な説明】

【図１】改良された最上位有効ビット予測器を備える加算機のブロック図である。

【図２】ＬＢＡ内の種々の構成要素、すなわちオリジナル・オペランド、シフトされたオペンド、ＰＫＧ記号、予測ビット値マップおよび未調整演算和のビット位置の間の対応関係を示す図式である。

【図３】図２に最下位有効ＰＫＧ記号を追加した図式である。

【図４】図１のＬＢＡに関連して使用されるエラー訂正の１つのタイプを示すブロック図である。

【図５】非ＭＳＢトリプルに関連して使用されるゲーティング・セルのＣＭＯＳ形態の図式である。

【図６】図２と図３と同様であるが、ＰＫＧ記号のＭＳＰペアの解読を含む図式である。

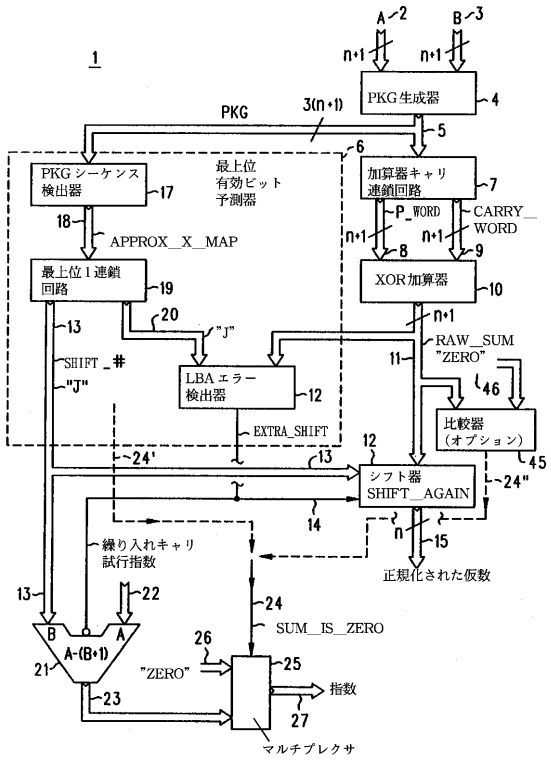
【図７】図６の形態に対して使用されるゲーティング・セルのＣＭＯＳ形態の図式である。

【符号の説明】

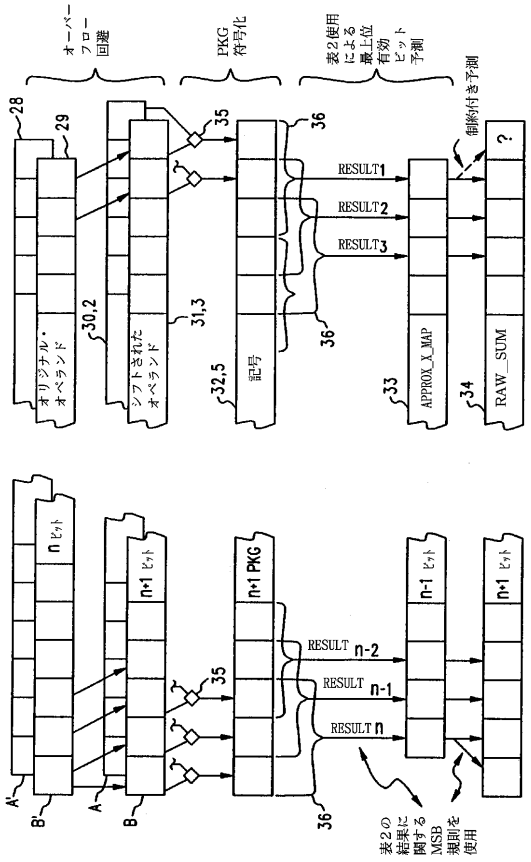
１　浮動小数点２進加算器
２　(ｎ+1)ビット仮数Ａ
３　(ｎ+1)ビット仮数Ｂ
４　ＰＫＧ生成器
５　(ｎ＋１)ビットＰＫＧワード
６　最上位有効ビット位置予測器(ＬＢＡ)
７　加算器キャリ連鎖回路
８　(ｎ＋１)P_WORD信号
９　(ｎ＋１)CARRY_WORD信号
１０　ＸＯＲ加算器
１１　RAW_SUM信号
１２　ＬＢＡエラー検出器
１３　SHIFT_#信号
１４　EXTRA_SHIFT信号
１５　正規化された仮数
１７　ＰＫＧシーケンス検出器
１８　PPROX_X_MAP信号
１９　最上位１連鎖回路
２０　値Ｊ
２１　加算器
２２　試行指数
２３　計算結果としての指数
２４　SUM_IS_ZERO信号
２５　マルチプレクサ
２６、４６　値ゼロ
２７　調整済み指数
２８、２９　ｎビットのオリジナル・オペランド
３０、３１　シフトされた(ｎ＋１)ビット・オペランド
３３、４１　(ｎ－１)個のAPPROX_X_MAP

３４　　(ｎ＋１)個のRAW_SUM
３５　　トリプルから結果を生成する規則
３６　　(ｎ＋１)個の記号
４２　　ＸＯＲゲート
４３、４４　　ＡＮＤゲート

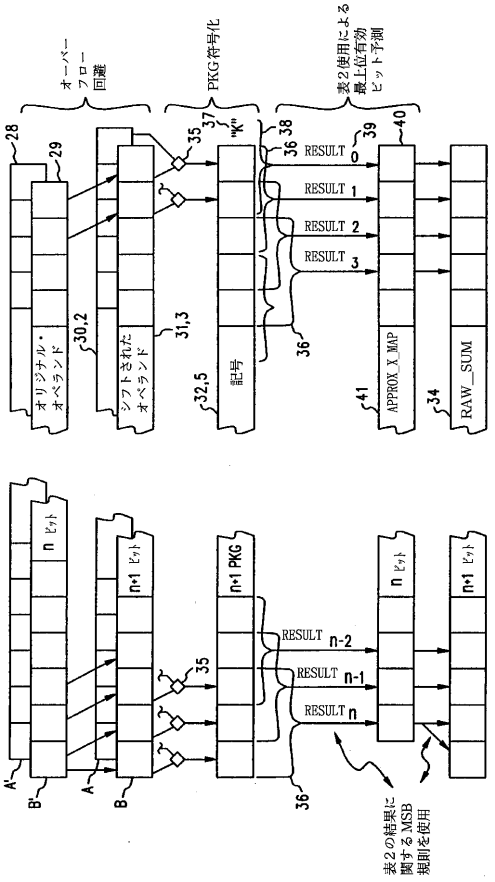
【図１】



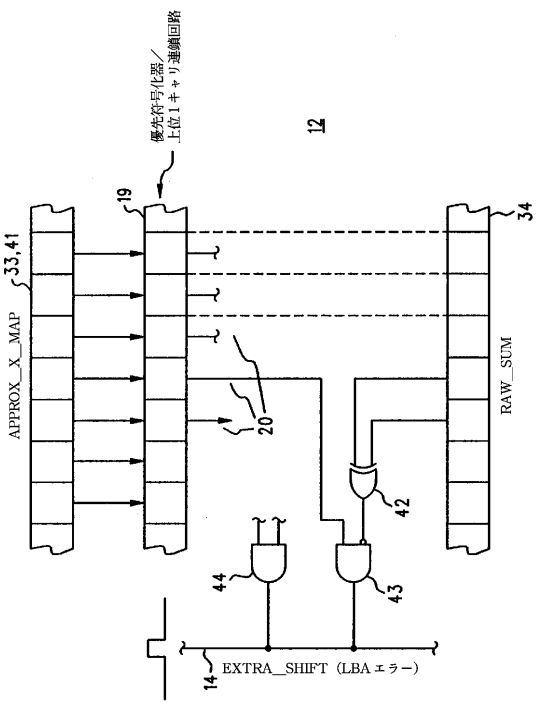
【図２】



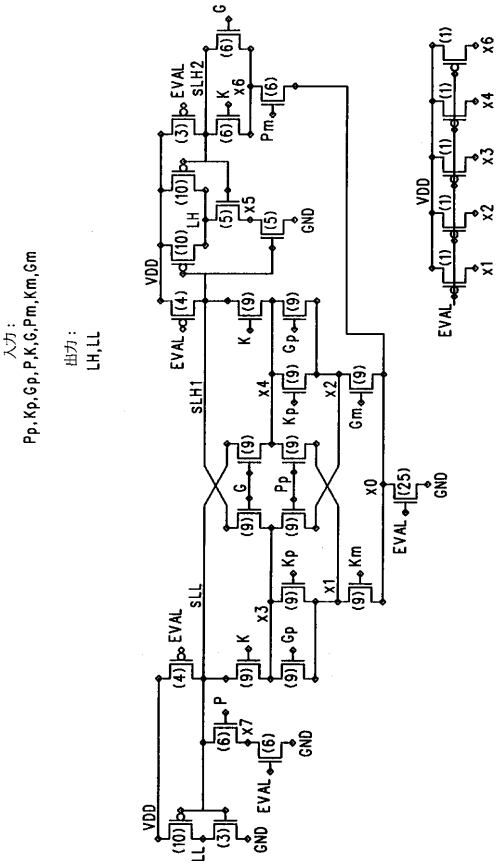
【図３】


オーバーフロー回避
PKG符号化
表2使用による最上位有効ビット予測
オリジナル・オペランド
シフトされたオペランド
記号
APPROX_X_MAP
RAW_SUM
RESULT 0
RESULT 1
RESULT 2
RESULT 3
RESULT n-2
RESULT n-1
RESULT n
表2の結果に関するMSB規則を使用

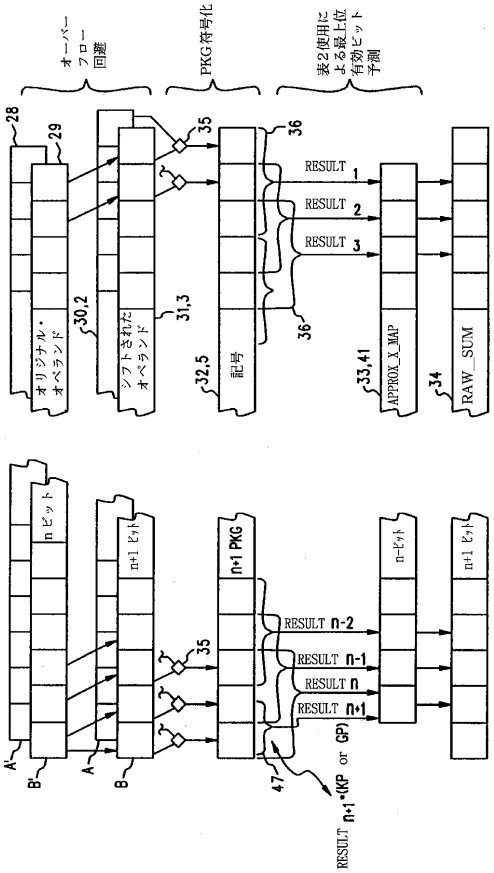
【図４】


優先符号化器／上位1キャリ連鎖回路
APPROX_X_MAP
RAW_SUM
EXTRA_SHIFT（LBAエラー）

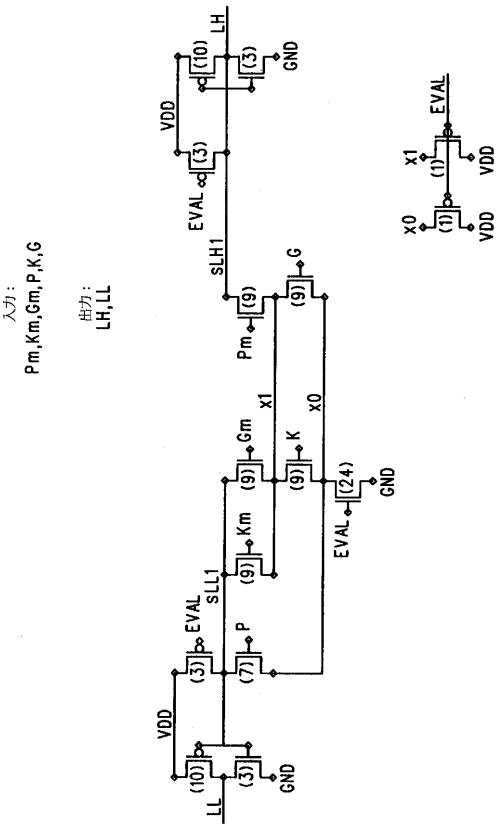
【図５】


入力：
Pp,Kp,Gp,P,K,G,Pm,Km,Gm
出力：
LH,LL
VDD
GND
EVAL
SLH1
SLH2
SLL
LH
LL

【図６】


オーバーフロー回避
PKG符号化
表2使用による最上位有効ビット予測
オリジナル・オペランド
シフトされたオペランド
記号
APPROX_X_MAP
RAW_SUM
RESULT 1
RESULT 2
RESULT 3
RESULT n-2
RESULT n-1
RESULT n
RESULT n+1
RESULT n+1*(KP or GP)

【図７】


入力：
Pm,Km,Gm,P,K,G
出力：
LH,LL
VDD
LH
(10)
(3)
GND
EVAL
(3)
SLH1
Pm
(9)
G
(9)
x1
Gm
(9)
K
(9)
x0
Km
(9)
(24)
GND
EVAL
SLL1
EVAL
(3)
P
(7)
VDD
(10)
(3)
GND
LL
X1
(1)
EVAL
VDD
X0
(1)
VDD

フロントページの続き

(72)発明者　ルドルフォ・ジー・バーハ
アメリカ合衆国９４０２４カリフォルニア州ロス・アルトス、ホーメステッド・コート　２２５０、アパートメント　３０４

審査官　田中　友章

(56)参考文献　特開昭６１－１７７５３８（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 7/00