(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2012-221189
(P2012-221189A)
(43)公開日　平成24年11月12日(2012.11.12)

(51)Int.Cl.	F I	テーマコード（参考）
G06F 7/57 (2006.01)	G06F 7/57	

審査請求 未請求　請求項の数 13　ＯＬ　（全 47 頁）

(21)出願番号　特願2011-85882(P2011-85882)
(22)出願日　平成23年4月8日(2011.4.8)

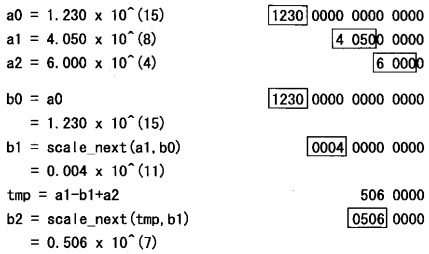
(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号
(74)代理人　100070150
弁理士　伊東　忠彦
(74)代理人　100146776
弁理士　山口　昭則
(72)発明者　菅　竜二
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
(72)発明者　海野　秀之
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

最終頁に続く

(54)【発明の名称】演算回路、演算処理装置、及び演算回路の制御方法

(57)【要約】
【課題】固定精度の浮動小数点数同士の正確な和を高速に実現する演算方式を提供する。
【解決手段】演算回路は、第１の仮数部と第１の指数部とを含む第１の加減算対象データを格納するレジスタと、第２の仮数部と第２の指数部とを含む第２の加減算対象データを格納するレジスタと、第１の指数部と第２の指数部との差の絶対値が、第１仮数部の先行ゼロ計数値以上であるか否かに応じて第１のシフト量及び第２のシフト量を算出するシフト量生成部と、第１の仮数部を第１のシフト量だけシフトし、第２の仮数部を前記第２のシフト量だけシフトするシフト部と、第１のシフト後仮数部と第２のシフト後仮数部とを加減算した倍精度による倍精度仮数部を出力する加減算部と、倍精度仮数部を正規化した正規化倍精度仮数部の下位側を正規化した下位側正規化仮数部と、正規化指数部から所定の固定精度の桁数だけ減算した下位側正規化指数部を算出する正規化部を有する。
【選択図】図２９

符号の揃ったPriestの再正規化された数の組から
強い正規化された数の組を得る計算の一例を示す図



【特許請求の範囲】
【請求項１】
２値を加減算した結果である所定の固定精度の範囲で表される加減算値に対する補正値を算出する演算回路において、
Ｎ進法（Ｎは２以上の整数）による第１の固定精度浮動小数点数の第１の仮数部と、前記第１の仮数部に対する指数を表す第１の指数部とを含む第１の形式の第１の加減算対象データを保持する第１の入力レジスタと、
前記Ｎ進法による第２の固定精度浮動小数点数の第２の仮数部と、前記第２の仮数部に対する指数を表す第２の指数部を含み、前記第２の指数部の値が前記第１の指数部の値より小さい第１の条件と、前記第１の指数部から前記第１の仮数部の最上位ビットから連続する０値を計数した結果である第１の先行ゼロ計数値を減算した第１の減算後指数部から、前記第２の指数部から前記第２の仮数部の最上位ビットから連続する０値を計数した結果である第２の先行ゼロ計数値を減算した第２の減算後指数部を減算した結果が、前記所定の固定精度の桁数より小さい値となる第２の条件とを同時に満たす第１の形式の第２の加減算対象データを保持する第２の入力レジスタと、
前記第１の指数部と前記第２の指数部の差の絶対値が、前記第１の先行ゼロ計数値以上である場合には前記第１の先行ゼロ計数値を、前記第１の先行ゼロ計数値よりも小さい場合には前記第１の指数部と前記第２の指数部の差の絶対値を、第１のシフト量として算出する第１のシフト量生成部と、
前記第１の指数部と前記第２の指数部の差の絶対値が、前記第１の先行ゼロ計数値以上である場合には前記第１の指数部から前記第２の指数部と前記第１の先行ゼロ計数値を減算した値を、前記第１の先行ゼロ計数値よりも小さい場合にはゼロを、第２のシフト量として算出する第２のシフト量生成部と、
前記第１の仮数部を、前記第１のシフト量だけシフトした第１のシフト後仮数部を算出する第１のシフト部と、
前記第２の仮数部を、前記第２のシフト量だけシフトした第２のシフト後仮数部を算出する第２のシフト部と、
前記第１のシフト後仮数部と前記第２のシフト後仮数部を加減算した前記所定の固定精度の倍精度による倍精度仮数部と、前記第１のシフト後仮数部と前記第２のシフト後仮数部の加減算によるキャリーを出力する加減算部と、
前記第１の指数部及び前記第２の指数部のうち小さい方の指数部に、前記第２のシフト量を加減算したシフト後指数部を算出するシフト後指数生成部と、
前記シフト後指数部に、前記キャリーを加減算した補正指数部を算出する補正指数生成部と、
前記補正指数部を正規化した正規化指数部と、前記倍精度仮数部を正規化した正規化倍精度仮数部とを算出する第１の正規化部と、
前記正規化倍精度仮数部の下位側の前記所定の固定精度の桁数分の仮数部を正規化した下位側正規化仮数部と、前記正規化指数部から前記所定の固定精度の桁数だけ減算した下位側正規化指数部を算出する第２の正規化部を有することを特徴とする演算回路。
【請求項２】
前記演算回路はさらに、
前記第１の入力レジスタに保持された第１の加減算対象データを第２の形式による第１の加減算対象データに変換する第１の変換部と、
前記第２の入力レジスタに保持された第２の加減算対象データを第２の形式による第２の加減算対象データに変換する第２の変換部と、
を有し、
前記第１のシフト量生成部は、前記第２の形式による第１の指数部と前記第２の形式による第２の指数部の差の絶対値が、前記第２の形式による第１の仮数部の最上位ビットから連続する０値を計数した結果である第１の先行ゼロ計数値以上である場合には前記第１の先行ゼロ計数値を、前記第１の先行ゼロ計数値よりも小さい場合には前記第２の形式に

よる第１の指数部と前記第２の形式による第２の指数部の差の絶対値を、第１のシフト量として算出し、
前記第２のシフト量生成部は、前記第２の形式による第１の指数部と前記第２の形式による第２の指数部の差の絶対値が、前記第１の先行ゼロ計数値以上である場合には前記第２の形式による第１の指数部から前記第２の指数部と前記第１の先行ゼロ計数値を減算した値を、前記第１の先行ゼロ計数値よりも小さい場合にはゼロを、第２のシフト量として算出し、
前記シフト後指数生成部は、前記第２の形式による第１の指数部及び前記第２の形式による第２の指数部のうち小さい方の指数部に、前記第２のシフト量を加減算したシフト後指数部を算出することを特徴とする請求項１記載の演算回路。
【請求項３】
前記演算回路において、
前記第２の形式は、２進化１０進数形式であることを特徴とする請求項２記載の演算回路。
【請求項４】
前記演算回路はさらに、
前記下位側正規化仮数部を前記第１の形式の下位側正規化仮数部に変換するとともに、前記下位側正規化指数部を前記第１の形式の下位側正規化指数部に変換する第３の変換部を有することを特徴とする請求項２又は３記載の演算回路。
【請求項５】
前記演算回路はさらに、
前記第１の加減算対象データと前記第２の加減算対象データとの指数部の大小を比較する比較回路と、
前記比較回路の比較結果に基づき、前記第１の加減算対象データと前記第２の加減算対象データのうち、指数部が大きい方の加減算対象データの仮数部を前記第１のシフト部のシフト対象として出力する第１の選択部と、
前記比較回路の比較結果に基づき、前記第１の加減算対象データと前記第２の加減算対象データのうち、指数部が小さい方の加減算対象データの仮数部を前記第２のシフト部のシフト対象として出力する第２の選択部を有することにより、前記第１の条件を満たす必要がないことを特徴とする請求項１～４のいずれか１項に記載の演算回路。
【請求項６】
前記演算回路において、
前記シフト後指数生成部はさらに、
前記第１の指数部から前記第１の先行ゼロ計数値を減算した第１の減算後指数部から、前記第２の指数部から前記第２の先行ゼロ計数値を減算した第２の減算後指数部を減算した結果が、前記所定の固定精度の桁数以上である場合には、前記第２の指数部と前記第２の仮数部を第２の正規化部の正規化対象とするバイパス回路を有することにより、前記第２の条件を満たす必要がないことを特徴とする請求項１～５のいずれか１項に記載の演算回路。
【請求項７】
２値を加減算した結果である所定の固定精度の範囲で表される加減算値に対する修正値を算出する演算回路と、前記２値を加減算した結果である所定の固定精度の範囲で表される加減算値に対する修正値を算出する修正値演算命令をデコードする命令制御部とを有する演算処理装置において、
前記演算回路は、
Ｎ進法（Ｎは２以上の整数）による第１の固定精度浮動小数点数の第１の仮数部と、前記第１の仮数部に対する指数を表す１つの第１の指数部を含む第１の形式の第１の加減算対象データを保持する第１の入力レジスタと、
前記Ｎ進法による第２の固定精度浮動小数点数の第２の仮数部と、前記第２の仮数部に対する指数を表す１つの第２の指数部を含み、前記第２の指数部が前記第１の指数部より

小さいことを前提とする第１の条件と、前記第１の指数部から前記第１の仮数部の最上位ビットから連続する０値を計数した結果である第１の先行ゼロ計数値を減算した第１の減算後指数部から、前記第２の指数部から前記第２の仮数部の最上位ビットから連続する０値を計数した結果である第２の先行ゼロ計数値を減算した第２の減算後指数部を減算した結果が、前記所定の固定精度の桁数より小さい値となることを前提とする第２の条件を同時に満たす第１の形式の第２の加減算対象データを保持する第２の入力レジスタと、

前記第１の指数部と前記第２の指数部の差の絶対値が、前記第１の先行ゼロ計数値以上である場合には前記第１の先行ゼロ計数値を、前記第１の先行ゼロ計数値よりも小さい場合には前記第１の指数部と前記第２の指数部の差の絶対値を、第１のシフト量として算出する第１のシフト量生成部と、

前記第１の指数部と前記第２の指数部の差の絶対値が、前記第１の先行ゼロ計数値以上である場合には前記第１の指数部から前記第２の指数部と前記第１の先行ゼロ計数値を減算した値を、前記第１の先行ゼロ計数値よりも小さい場合にはゼロを、第２のシフト量として算出する第２のシフト量生成部と、

前記第１の仮数部を、前記第１のシフト量だけシフトした第１のシフト後仮数部を算出する第１のシフト部と、

前記第２の仮数部を、前記第２のシフト量だけシフトした第２のシフト後仮数部を算出する第２のシフト部と、

前記第１のシフト後仮数部と前記第２のシフト後仮数部を加減算した前記所定の固定精度の倍精度による倍精度仮数部と、前記加減算によるキャリーを出力する加減算部と、

前記第１の指数部及び前記第２の指数部のうち小さい方の指数部に、前記第２のシフト量を加減算したシフト後指数部を算出するシフト後指数生成部と、

前記シフト後指数部に、前記キャリーを加減算した補正指数部を算出する補正指数生成部と、

前記補正指数部を正規化した正規化指数部と、前記倍精度仮数部を正規化した正規化倍精度仮数部とを算出する第１の正規化部と、

前記正規化倍精度仮数部の下位側の前記所定の固定精度による仮数部を正規化した下位側正規化仮数部と、前記正規化指数部から前記所定の固定精度の桁数だけ減算した下位側正規化指数部を算出する第２の正規化部を有することを特徴とする演算処理装置。

【請求項８】

前記演算処理装置の演算回路はさらに、

前記第１の入力レジスタに保持された第１の加減算対象データを第２の形式による第１の加減算対象データに変換する第１の変換部と、

前記第２の入力レジスタに保持された第２の加減算対象データを第２の形式による第２の加減算対象データに変換する第２の変換部と、

を有し、

前記第１のシフト量生成部は、前記第２の形式による第１の指数部と前記第２の形式による第２の指数部の差の絶対値が、前記第２の形式による第１の仮数部の最上位ビットから連続する０値を計数した結果である第１の先行ゼロ計数値以上である場合には前記第１の先行ゼロ計数値を、前記第１の先行ゼロ計数値よりも小さい場合には前記第２の形式による第１の指数部と前記第２の形式による第２の指数部の差の絶対値を、第１のシフト量として算出し、

前記第２のシフト量生成部は、前記第２の形式による第１の指数部と前記第２の形式による第２の指数部の差の絶対値が、前記第１の先行ゼロ計数値以上である場合には前記第２の形式による第１の指数部から前記第２の指数部と前記第１の先行ゼロ計数値を減算した値を、前記第１の先行ゼロ計数値よりも小さい場合にはゼロを、第２のシフト量として算出し、

前記シフト後指数生成部は、前記第２の形式による第１の指数部及び前記第２の形式による第２の指数部のうち小さい方の指数部に、前記第２のシフト量を加減算したシフト後指数部を算出することを特徴とする請求項７記載の演算処理装置。

【請求項９】
前記演算処理装置の演算回路において、
前記第２の形式は、２進化１０進数形式であることを特徴とする請求項８記載の演算処理装置。
【請求項１０】
前記演算処理装置の演算回路はさらに、
前記下位側正規化仮数部を前記第１の形式の下位側正規化仮数部に変換するとともに、前記下位側正規化指数部を前記第１の形式の下位側正規化指数部に変換する第３の変換部を有することを特徴とする請求項８又は９のいずれか１項に記載の演算処理装置。
【請求項１１】
前記演算処理装置の前記演算回路はさらに、
前記第１の加減算対象データと前記第２の加減算対象データとの指数部の大小を比較する比較回路と、
前記比較回路の比較結果に基づき、前記第１の加減算対象データと前記第２の加減算対象データのうち、指数部が大きい方の加減算対象データの仮数部を前記第１のシフト部のシフト対象として出力する第１の選択部と、
前記比較回路の比較結果に基づき、前記第１の加減算対象データと前記第２の加減算対象データのうち、指数部が小さい方の加減算対象データの仮数部を前記第２のシフト部のシフト対象として出力する第２の選択部を有することを特徴とする請求項７～１０のいずれか１項に記載の演算処理装置。
【請求項１２】
前記演算処理装置の前記演算回路において、
前記シフト後指数生成部はさらに、
前記第１の指数部から前記第１の先行ゼロ計数値を減算した第１の減算後指数部から、前記第２の指数部から前記第２の先行ゼロ計数値を減算した第２の減算後指数部を減算した結果が、前記所定の固定精度の桁数以上である場合には、前記第２の指数部と前記第２の仮数部を第２の正規化部の正規化対象とするバイパス回路を有することを特徴とする請求項７～１１のいずれか１項に記載の演算処理装置。
【請求項１３】
Ｎ進法（Ｎは２以上の整数）による第１の固定精度浮動小数点数の第１の仮数部と、前記第１の仮数部に対する指数を表す１つの第１の指数部を含む第１の形式の第１の加減算対象データを保持する第１の入力レジスタと、前記Ｎ進法による第２の固定精度浮動小数点数の第２の仮数部と、前記第２の仮数部に対する指数を表す１つの第２の指数部を含み、前記第２の指数部が前記第１の指数部より小さいことを前提とする第１の条件と、前記第１の指数部から前記第１の仮数部の最上位ビットから連続する０値を計数した結果である第１の先行ゼロ計数値を減算した第１の減算後指数部から、前記第２の指数部から前記第２の仮数部の最上位ビットから連続する０値を計数した結果である第２の先行ゼロ計数値を減算した第２の減算後指数部を減算した結果が、前記所定の固定精度の桁数より小さい値となることを前提とする第２の条件を同時に満たす第１の形式の第２の加減算対象データを保持する第２の入力レジスタを有するとともに、前記第１の加減算対象データと第２の加減算対象データを加減算した結果である所定の固定精度の範囲で表される加減算値に対する補正値を算出する演算回路の制御方法において、
前記演算回路が有する第１のシフト量生成部が、前記第１の先行ゼロ計数値以上である場合には前記第１の先行ゼロ計数値を、前記第１の先行ゼロ計数値よりも小さい場合には前記第１の指数部と前記第２の指数部の差の絶対値を、第１のシフト量として算出し、
前記演算回路が有する第２のシフト量生成部が、前記第１の指数部と前記第２の指数部の差の絶対値が、前記第１の先行ゼロ計数値以上である場合には前記第１の指数部から前記第２の指数部と前記第１の先行ゼロ計数値を減算した値を、前記第１の先行ゼロ計数値よりも小さい場合にはゼロを、第２のシフト量として算出し、
前記演算回路が有する第１のシフト部が、前記第１の仮数部を、前記第１のシフト量だ

けシフトした第１のシフト後仮数部を算出し、
前記演算回路が有する第２のシフト部が、前記第２の仮数部を、前記第２のシフト量だけシフトした第２のシフト後仮数部を算出し、
前記演算回路が有する加減算部が、前記第１のシフト後仮数部と前記第２のシフト後仮数部を加減算した前記所定の固定精度の倍精度による倍精度仮数部と、前記加減算によるキャリーを出力し、
前記演算回路が有するシフト後指数生成部が、前記第１の指数部及び前記第２の指数部のうち小さい方の指数部に、前記第２のシフト量を加減算したシフト後指数部を算出し、
前記シフト後指数部に、前記キャリーを加減算した補正指数部を算出する補正指数生成部と、
前記演算回路が有する第１の正規化部が、前記補正指数部を正規化した正規化指数部と、前記倍精度仮数部を正規化した正規化倍精度仮数部とを算出し、
前記演算回路が有する第２の正規化部が、前記正規化倍精度仮数部の下位側の前記所定の固定精度による仮数部を正規化した下位側正規化仮数部と、前記正規化指数部から前記所定の固定精度の桁数だけ減算した下位側正規化指数部を算出することを特徴とする演算回路の制御方法。

【発明の詳細な説明】

【技術分野】

【０００１】

本願開示は、演算回路、演算処理装置、及び演算回路の制御方法に関する。

【背景技術】

【０００２】

銀行等において勘定処理を行なう勘定系や、一部の科学技術計算などでは、数値表現や演算における誤差を小さくするために、多倍長、または、可変長の数値表現形式が採用されることがある。このような場合、符号と指数とをひとつの整数で表現し、仮数については、ひとつの整数で表現した符号と指数とは別の整数の列で表現することが多い。また、このような数値表現を採用した場合には、整数演算を利用して数値同士の計算が実現されることも多くあった。

【０００３】

それに対し、多倍長や可変長の浮動小数点演算を、固定精度の浮動小数点演算を用いて実現する方法が提案されている。固定精度の浮動小数点演算については、ハードウエアによる処理手段を提供している場合が多いため、そのようなハードウエア処理手段を用いることで、全ての処理をソフトウェアに行なわせる場合よりも処理の高速化をはかることができる。例えば、多倍長の２進浮動小数点演算を、２倍長の浮動小数点演算を用いて行うライブラリも存在する。これらの方法では、固定精度浮動小数点数の組（要素同士の足し算を実行せずに組のまま用いることから「未評価の和」とも呼ばれる）によりひとつの数を表現し、その組同士の算術を適切に実行することで、大きな精度の算術（四則演算）を実現している。

【０００４】

固定精度の浮動小数点演算においては、演算結果は必ずしも正確にはならない。たとえば、固定精度の浮動小数点数同士の正確な和を実現するためには、計算結果を２つの固定精度の浮動小数点数で表現する必要がある。このような正確な和のアルゴリズムは、次のようになる。

【０００５】

```
Two_sum(X, Y)
  z  = fl(X + Y)
  w  = fl(z - X)
  v  = fl(z - w)
  z1 = fl(Y - w)
  z2 = fl(v - X)
```

```
    zz = fl(z1 - z2)
    return(z, zz)
```

ここでｆｌ（Ｘ＋Ｙ）は、Ｘ＋Ｙの真の値を、浮動小数点数にマッピングした結果、即ち浮動小数点形式の限られた精度内で表現した結果を示す。上記のＴｗｏ＿ｓｕｍによって得られる２数ｚ及びｚｚは、正確な意味においてｚ＋ｚｚ＝Ｘ＋Ｙを満たす。ｚはＸ＋Ｙの最重要部（most significant part）を固定精度浮動小数点の精度内で表現した値であり、ｚｚは固定精度浮動小数点の精度で表現しきれなかった残余分を表す。

【０００６】

マッピング時に発生する丸め処理方法として、１０進数で代表的な丸め方法である四捨五入を例に取り説明する。簡単のために固定精度浮動小数点数の精度は１０進２桁とする。この場合、２００００と－１との２数の和は、上記のＴｗｏ＿ｓｕｍにより以下のように計算される。

【０００７】

```
  X = 20000
  Y =    -1
  z = fl(X＋Y) = 20000
  w = fl(z－X) = 0
  v = fl(z－w) = 20000
  z1 = fl(Y－w) = -1
  z2 = fl(v－X) = 0
  zz = fl(z1－z2) = -1
```

この計算により、正確な意味においてｚ＋ｚｚ＝Ｘ＋Ｙであるようなｚ及びｚｚを求めることができる。

【０００８】

上記のアルゴリズムでは、ひとつの和ｚ＋ｚｚを実現するために、６回もの浮動小数点演算が必要となる。この正確な和アルゴリズムは、多倍長や可変長の浮動小数点演算を、固定精度の浮動小数点演算を用いて実現する場合に頻繁に用いられるものである。従って、この正確な和アルゴリズムを高速化することが望まれる。

【先行技術文献】

【非特許文献】

【０００９】

【非特許文献１】T. Dekker, A Floating-Point Technique for Extending the Available Precision, Numer. Math. vol. 18, pp.224-242, 1971.

【非特許文献２】D. Priest, Appendix A: Algorithms for Arbitrary Precision Floating Point Arithmetic, pp.111-124, On Property of Floating Point Arithmetics: Numerical Stability and the Cost of Accurate Computations, PhD thesis, University of California, Berkeley, November 1992.

【非特許文献３】Yozo Hida, Xiaoye S. Li, David H. Bailey, Library for Double-Double and Quad-Double Arithmetic, 29 December 2007.

【発明の概要】

【発明が解決しようとする課題】

【００１０】

以上を鑑みると、固定精度の浮動小数点数同士の正確な和を高速に実現する演算方式が望まれる。

【課題を解決するための手段】

【００１１】

２値を加減算した結果である所定の固定精度の範囲で表される加減算値に対する補正値を算出する演算回路は、Ｎ進法（Ｎは２以上の整数）による第１の固定精度浮動小数点数の第１の仮数部と、前記第１の仮数部に対する指数を表す第１の指数部とを含む第１の形式の第１の加減算対象データを保持する第１の入力レジスタと、前記Ｎ進法による第２の

固定精度浮動小数点数の第２の仮数部と、前記第２の仮数部に対する指数を表す第２の指数部を含み、前記第２の指数部の値が前記第１の指数部の値より小さい第１の条件と、前記第１の指数部から前記第１の仮数部の最上位ビットから連続する０値を計数した結果である第１の先行ゼロ計数値を減算した第１の減算後指数部から、前記第２の指数部から前記第２の仮数部の最上位ビットから連続する０値を計数した結果である第２の先行ゼロ計数値を減算した第２の減算後指数部を減算した結果が、前記所定の固定精度の桁数より小さい値となる第２の条件とを同時に満たす第１の形式の第２の加減算対象データを保持する第２の入力レジスタと、前記第１の指数部と前記第２の指数部の差の絶対値が、前記第１の先行ゼロ計数値以上である場合には前記第１の先行ゼロ計数値を、前記第１の先行ゼロ計数値よりも小さい場合には前記第１の指数部と前記第２の指数部の差の絶対値を、第１のシフト量として算出する第１のシフト量生成部と、前記第１の指数部と前記第２の指数部の差の絶対値が、前記第１の先行ゼロ計数値以上である場合には前記第１の指数部から前記第２の指数部と前記第１の先行ゼロ計数値を減算した値を、前記第１の先行ゼロ計数値よりも小さい場合にはゼロを、第２のシフト量として算出する第２のシフト量生成部と、前記第１の仮数部を、前記第１のシフト量だけシフトした第１のシフト後仮数部を算出する第１のシフト部と、前記第２の仮数部を、前記第２のシフト量だけシフトした第２のシフト後仮数部を算出する第２のシフト部と、前記第１のシフト後仮数部と前記第２のシフト後仮数部を加減算した前記所定の固定精度の倍精度による倍精度仮数部と、前記第１のシフト後仮数部と前記第２のシフト後仮数部の加減算によるキャリーを出力する加減算部と、前記第１の指数部及び前記第２の指数部のうち小さい方の指数部に、前記第２のシフト量を加減算したシフト後指数部を算出するシフト後指数生成部と、前記シフト後指数部に、前記キャリーを加減算した補正指数部を算出する補正指数生成部と、前記補正指数部を正規化した正規化指数部と、前記倍精度仮数部を正規化した正規化倍精度仮数部とを算出する第１の正規化部と、前記正規化倍精度仮数部の下位側の前記所定の固定精度の桁数分の仮数部を正規化した下位側正規化仮数部と、前記正規化指数部から前記所定の固定精度の桁数だけ減算した下位側正規化指数部を算出する第２の正規化部を有することを特徴とする。

【００１２】

２値を加減算した結果である所定の固定精度の範囲で表される加減算値に対する修正値を算出する演算回路と、前記２値を加減算した結果である所定の固定精度の範囲で表される加減算値に対する修正値を算出する修正値演算命令をデコードする命令制御部とを有する演算処理装置において、前記演算回路は、Ｎ進法（Ｎは２以上の整数）による第１の固定精度浮動小数点数の第１の仮数部と、前記第１の仮数部に対する指数を表す１つの第１の指数部を含む第１の形式の第１の加減算対象データを保持する第１の入力レジスタと、前記Ｎ進法による第２の固定精度浮動小数点数の第２の仮数部と、前記第２の仮数部に対する指数を表す１つの第２の指数部を含み、前記第２の指数部が前記第１の指数部より小さいことを前提とする第１の条件と、前記第１の指数部から前記第１の仮数部の最上位ビットから連続する０値を計数した結果である第１の先行ゼロ計数値を減算した第１の減算後指数部から、前記第２の指数部から前記第２の仮数部の最上位ビットから連続する０値を計数した結果である第２の先行ゼロ計数値を減算した第２の減算後指数部を減算した結果が、前記所定の固定精度の桁数より小さい値となることを前提とする第２の条件を同時に満たす第１の形式の第２の加減算対象データを保持する第２の入力レジスタと、前記第１の指数部と前記第２の指数部の差の絶対値が、前記第１の先行ゼロ計数値以上である場合には前記第１の先行ゼロ計数値を、前記第１の先行ゼロ計数値よりも小さい場合には前記第１の指数部と前記第２の指数部の差の絶対値を、第１のシフト量として算出する第１のシフト量生成部と、前記第１の指数部と前記第２の指数部の差の絶対値が、前記第１の先行ゼロ計数値以上である場合には前記第１の指数部から前記第２の指数部と前記第１の先行ゼロ計数値を減算した値を、前記第１の先行ゼロ計数値よりも小さい場合にはゼロを、第２のシフト量として算出する第２のシフト量生成部と、前記第１の仮数部を、前記第１のシフト量だけシフトした第１のシフト後仮数部を算出する第１のシフト部と、前記第

２の仮数部を、前記第２のシフト量だけシフトした第２のシフト後仮数部を算出する第２のシフト部と、前記第１のシフト後仮数部と前記第２のシフト後仮数部を加減算した前記所定の固定精度の倍精度による倍精度仮数部と、前記加減算によるキャリーを出力する加減算部と、前記第１の指数部及び前記第２の指数部のうち小さい方の指数部に、前記第２のシフト量を加減算したシフト後指数部を算出するシフト後指数生成部と、前記シフト後指数部に、前記キャリーを加減算した補正指数部を算出する補正指数生成部と、前記補正指数部を正規化した正規化指数部と、前記倍精度仮数部を正規化した正規化倍精度仮数部とを算出する第１の正規化部と、前記正規化倍精度仮数部の下位側の前記所定の固定精度による仮数部を正規化した下位側正規化仮数部と、前記正規化指数部から前記所定の固定精度の桁数だけ減算した下位側正規化指数部を算出する第２の正規化部を有することを特徴とする。

【００１３】

Ｎ進法（Ｎは２以上の整数）による第１の固定精度浮動小数点数の第１の仮数部と、前記第１の仮数部に対する指数を表す１つの第１の指数部を含む第１の形式の第１の加減算対象データを保持する第１の入力レジスタと、前記Ｎ進法による第２の固定精度浮動小数点数の第２の仮数部と、前記第２の仮数部に対する指数を表す１つの第２の指数部を含み、前記第２の指数部が前記第１の指数部より小さいことを前提とする第１の条件と、前記第１の指数部から前記第１の仮数部の最上位ビットから連続する０値を計数した結果である第１の先行ゼロ計数値を減算した第１の減算後指数部から、前記第２の指数部から前記第２の仮数部の最上位ビットから連続する０値を計数した結果である第２の先行ゼロ計数値を減算した第２の減算後指数部を減算した結果が、前記所定の固定精度の桁数より小さい値となることを前提とする第２の条件を同時に満たす第１の形式の第２の加減算対象データを保持する第２の入力レジスタを有するとともに、前記第１の加減算対象データと第２の加減算対象データを加減算した結果である所定の固定精度の範囲で表される加減算値に対する補正値を算出する演算回路の制御方法において、前記演算回路が有する第１のシフト量生成部が、前記第１の先行ゼロ計数値以上である場合には前記第１の先行ゼロ計数値を、前記第１の先行ゼロ計数値よりも小さい場合には前記第１の指数部と前記第２の指数部の差の絶対値を、第１のシフト量として算出し、前記演算回路が有する第２のシフト量生成部が、前記第１の指数部と前記第２の指数部の差の絶対値が、前記第１の先行ゼロ計数値以上である場合には前記第１の指数部から前記第２の指数部と前記第１の先行ゼロ計数値を減算した値を、前記第１の先行ゼロ計数値よりも小さい場合にはゼロを、第２のシフト量として算出し、前記演算回路が有する第１のシフト部が、前記第１の仮数部を、前記第１のシフト量だけシフトした第１のシフト後仮数部を算出し、前記演算回路が有する第２のシフト部が、前記第２の仮数部を、前記第２のシフト量だけシフトした第２のシフト後仮数部を算出し、前記演算回路が有する加減算部が、前記第１のシフト後仮数部と前記第２のシフト後仮数部を加減算した前記所定の固定精度の倍精度による倍精度仮数部と、前記加減算によるキャリーを出力し、前記演算回路が有するシフト後指数生成部が、前記第１の指数部及び前記第２の指数部のうち小さい方の指数部に、前記第２のシフト量を加減算したシフト後指数部を算出し、前記シフト後指数部に、前記キャリーを加減算した補正指数部を算出する補正指数生成部と、前記演算回路が有する第１の正規化部が、前記補正指数部を正規化した正規化指数部と、前記倍精度仮数部を正規化した正規化倍精度仮数部とを算出し、前記演算回路が有する第２の正規化部が、前記正規化倍精度仮数部の下位側の前記所定の固定精度による仮数部を正規化した下位側正規化仮数部と、前記正規化指数部から前記所定の固定精度の桁数だけ減算した下位側正規化指数部を算出することを特徴とする。

【発明の効果】

【００１４】

本願開示の少なくとも１つの実施例によれば、固定精度の浮動小数点数同士の正確な和を高速に実現する演算方式が提供される。

【図面の簡単な説明】

【００１５】
【図１】Ｏｒａｃｌｅ－ｎｕｍｂｅｒの具体例の表を示す図である。
【図２】コンピュータシステムの構成の一例を示す図である。
【図３】ｏｒａｃｌｅｎｕｍ６４の構成を示す図である。
【図４】長さが９バイトのｏｒａｃｌｅｎｕｍ６４の構成を示す図である。
【図５】ｏｒａｃｌｅｎｕｍ６４を直接演算対象とできる演算器の構成の一例である。
【図６】指数仮数演算回路の構成の一例を示す図である。
【図７】正規化回路の構成の一例を示す図である。
【図８】先行ゼロ計数回路の構成の一例を示す図である。
【図９】内部形式変換回路の構成の一例を示す図である。
【図１０】外部形式変換回路の構成の一例を示す図である。
【図１１】固定精度浮動小数点数加算の具体例を示す図である。
【図１２】固定精度浮動小数点数加算の別の具体例を示す図である。
【図１３】２１バイト長のＯｒａｃｌｅ－ｎｕｍｂｅｒを３つの部分に分ける方法を示す図である。
【図１４】分割された３つの仮数部にそれぞれ対応するｏｒａｃｌｅｎｕｍ６４数を生成する方法を示す図である。
【図１５】ｇｅｔ＿ｚ演算を実現する回路の構成の一例を示す図である。
【図１６】ｇｅｔ＿ｚ演算の具体例を示す図である。
【図１７】ｇｅｔ＿ｚ演算の別の具体例を示す図である。
【図１８】ｇｅｔ＿ｚ演算の更に別の具体例を示す図である。
【図１９】２つの入力値の絶対値がフォーマットの桁数以上離れている場合の加算演算の例を示す図である。
【図２０】ｇｅｔ＿ｚｚ演算を実現する回路の構成の一例を示す図である。
【図２１】ｇｅｔ＿ｚｚ演算の具体例を示す図である。
【図２２】ｇｅｔ＿ｚｚ演算の別の具体例を示す図である。
【図２３】ｇｅｔ＿ｚｚ演算の更に別の具体例を示す図である。
【図２４】ｔｗｏ＿ｓｕｍの回路図シンボルを示す図である。
【図２５】ｔｒｉｐｌｅ－ｏｒａｃｌｅｎｕｍ６４とｏｒａｃｌｅｎｕｍ６４の和を求める回路の一例を示す図である。
【図２６】ｔｒｉｐｌｅ－ｏｒａｃｌｅｎｕｍ６４同士の和を求めるアルゴリズムを示す図である。
【図２７】オーバーラップを除去するための３つのｔｗｏ＿ｓｕｍ演算子の接続を示す図である。
【図２８】符号の揃ったＰｒｉｅｓｔの再正規化の例を示す図である。
【図２９】符号の揃ったＰｒｉｅｓｔの再正規化された数の組から強い正規化された数の組を得る計算の一例を示す図である。
【図３０】量子化のためのｓｃａｌｅ＿ｎｅｘｔ（Ｘ，Ｙ）演算を実現する回路構成の一例を示す図である。
【図３１】仮数部を内部形式に変換せずにｓｃａｌｅ＿ｎｅｘｔ演算を行う回路の構成の一例を示す図である。
【図３２】Ｏｒａｃｌｅ－ＤａｔａｂａｓｅにおけるＮＵＭＢＥＲ型の３種類の精度指定方法を示す表である。
【図３３】四捨五入が発生する桁位置のみ５があるような数を生成するアルゴリズムの例を示す図である。
【図３４】ｇｅｔ＿ｃｏｍｍａ５演算を実現する回路構成の一例を示す図である。
【図３５】ｔｒｕｎｃａｔｅ演算を実現する回路構成の一例を示す図である。
【図３６】マスク値生成回路の構成の一例を示す図である。
【図３７】ｐ＿ｄ及びｐ＿ｓの大小比較エラー判定回路の構成の一例を示す図である。
【図３８】符号演算器の回路構成の一例を示す図である。

【図３９】Ｏｒａｃｌｅ－ｎｕｍｂｅｒの長さを求める演算の一例を示す図である。
【図４０】ｅｘｐａｎｄ演算を実現する回路の構成の一例を示す図である。
【図４１】後続ゼロ計数回路の構成の一例を示す図である。
【図４２】桁セレクト演算回路の構成の一例を示す図である。
【図４３】固定精度浮動小数点加減算器の構成の一例を示す図である。
【図４４】指数仮数マスク演算回路の構成の一例を示す図である。
【発明を実施するための形態】
【００１６】
以下に、本発明の実施例を添付の図面を用いて詳細に説明する。なお各図面において、同一又は対応する構成要素は同一又は対応する名称又は番号で参照し、その説明は適宜省略する。
【００１７】
以下に説明する実施例では、オラクルデータベース（商標）で用いられる数値型であるＯｒａｃｌｅ－ｎｕｍｂｅｒ（商標）を数値表現の一例として用い、これを高速に計算するハードウエアが提供される。まず、Ｏｒａｃｌｅ－ｎｕｍｂｅｒの表現形式について述べる。ここで述べる事柄は、オラクルデータベースのＳＱＬインタプリタ（Ｓｔｒｕｃｔｕｒｅｄ Ｑｕｅｒｙ Ｌａｎｇｕａｇｅ ｉｎｔｅｒｐｒｅｔｅｒ）上で確認することができる。
【００１８】
Ｏｒａｃｌｅ－ｎｕｍｂｅｒは、最大２１バイトの可変長データ形式である。第１バイトに符号及び指数が格納され、後続バイトに仮数が格納される。仮数部は最大で２０バイトである。
【００１９】
Ｏｒａｃｌｅ－ｎｕｍｂｅｒは、１０進浮動小数点数を表現するためのデータ形式である。主にメモリ使用効率上の理由から、仮数部は、１バイトあたり１０進２桁分のデータを保持する。またそれにあわせて、指数部は、基数１００に対する指数が格納される。Ｏｒａｃｌｅ－ｎｕｍｂｅｒで表現される数は、以下の形式で表すことができる。
【００２０】
number = ± （M00 . M01 M02 ...) * 100^(exp)
ここでＭ００、Ｍ０１、Ｍ０２、・・・は、最大２０バイトで表現される仮数部における第１バイト、第２バイト、第３バイト、・・・の各バイトの格納データを示す。仮数部は１０進２桁毎に区切られているため、１００進２０桁とみなすこともできる。Ｏｒａｃｌｅ－ｎｕｍｂｅｒは、この１００進表現でみなしたときに必ず正規化されており、Ｍ００部（仮数部第１バイト）が０になることはない。
【００２１】
Ｏｒａｃｌｅ－ｎｕｍｂｅｒ表現の第１バイト（全体の第１バイト）は、符号及び指数部であり、次のように符号化される。
【００２２】
number > 0 の場合：第１バイト = exp + 193
number == 0 の場合：第１バイト = 128
それ以外の場合：第１バイト = 62 - exp
第２バイト以降の仮数部は、バイト毎にＭ００、Ｍ０１、・・・を保持する。各バイトにおいては、表現する数値の符号に応じて、以下に示すように異なった符号化がもちいられる。
【００２３】
number > 0 の場合：仮数部の第ｎバイト = M(n-1) + 1
number == 0 の場合：仮数部はない
それ以外の場合：仮数部の第ｎバイト = 101 - M(n-1)
上記の符号化において、Ｍｎは０以上且つ９９以下であるので、仮数部バイトの値に０Ｘ００が出現することがない。表現対象の数が短い仮数部で表現可能な場合には、Ｏｒａｃ

ｌｅ－ｎｕｍｂｅｒは２１バイトよりも短く切り詰められる。即ち、Ｏｒａｃｌｅ－ｎｕｍｂｅｒの仮数部には後続ゼロは許されていない。なお負数の表現においては、仮数部が２０バイトよりも短い場合には、仮数の終端を示すため、最後のバイトにターミネータとして１０２（０Ｘ６６）が格納される。

【００２４】

Ｏｒａｃｌｅ－ｎｕｍｂｅｒでは、上記のような符号化方式を採用することで、バイト列としてみたときの大小関係、即ちＣ標準関数ｍｅｍｃｍｐによる比較に基づく大小関係と、Ｏｒａｃｌｅ－ｎｕｍｂｅｒの数値としての大小関係とが等しくなる。

【００２５】

図１は、Ｏｒａｃｌｅ－ｎｕｍｂｅｒの具体例の表を示す図である。例えば１０Ｅ＋０（＝１０×１００^{0}）の表現において、指数部は１９３（＝０＋１９３）であり、仮数部は１１（＝１０＋１）である。また例えば１０Ｅ＋１（＝１×１００^{1}）の表現において、指数部は１９４（＝１＋１９３）であり、仮数部は２（＝１＋１）である。また例えば負の数－１０Ｅ－１３０（＝－１０×１００$^{-65}$）の表現において、指数部は１２７（＝６２－（－６５））であり、仮数部は９１（＝１０１－１０）である。また例えば負の数－１０Ｅ－１２９（＝－１×１００$^{-64}$）の表現において、指数部は１２６（＝６２－（－６４））であり、仮数部は１００（＝１０１－１）である。なお負の数には、最後のバイトにターミネータとして１０２が付加されている。更に、正の無限大Ｉｎｆ及び負の無限大－Ｉｎｆには、表中に示すような特別なバイト列が割り当てられている。

【００２６】

図２は、コンピュータシステムの構成の一例を示す図である。図２に示すコンピュータシステムは、プロセッサ１１０及びメモリ１１１を含む。演算処理装置としてのプロセッサ１１０は、２次キャッシュ部１１２、１次キャッシュ部１１３、制御部１１４、及び演算部１１５を含む。１次キャッシュ部１１３は、命令キャッシュ１１３Ａ及びデータキャッシュ１１３Ｂを含む。演算部１１５は、レジスタ１１６、演算制御部１１７、及び演算器１１８を含む。演算器１１８には演算回路１１９が含まれる。なお図２及び以降の同様の図において、各ボックスで示される各機能ブロックと他の機能ブロックとの境界は、基本的には機能的な境界を示すものであり、物理的な位置の分離、電気的な信号の分離、制御論理的な分離等に対応するとは限らない。各機能ブロックは、他のブロックと物理的にある程度分離された１つのハードウェアモジュールであってもよいし、或いは他のブロックと物理的に一体となったハードウェアモジュール中の１つの機能を示したものであってもよい。各機能ブロックは、他のブロックと論理的にある程度分離された１つのモジュールであってもよいし、或いは他のブロックと論理的に一体となったモジュール中の１つの機能を示したものであってもよい。

【００２７】

上記コンピュータシステムはＣＰＵ（Ｃｅｎｔｒａｌ　Ｐｒｏｃｅｓｓｉｎｇ　Ｕｎｉｔ）を用いた情報処理装置を模式化したものであり、このコンピュータシステムによりＯｒａｃｌｅ－ｎｕｍｂｅｒを演算するハードウエアを実現する。その際、システム構成を大幅に変更することなく、新機能を演算部１１５の機能として追加することが望まれる。そこで、新機能追加による変更部分が可能な限り少なくなるような実現方法をめざす。例えば、現状のＣＰＵにおいては、演算器の入出力は、通常２オペランド入力且つ１出力の形であり、各オペランドのデータ幅は８バイト（６４ビット）幅である。ハードウエアの変更量を小さく抑えるためには、この構造を大きく変えないことが望まれる。

【００２８】

プロセッサ１１０では、１次キャッシュ部１１３及び２次キャッシュ部１１２を設けることにより、キャッシュメモリを多階層化した構成となっている。具体的には、１次キャッシュ部１１３と主記憶（メモリ１１１）との間に、主記憶よりも高速にアクセスできる２次キャッシュ部１１２を設けている。これにより、１次キャッシュ部１１３においてキャッシュミスが発生した場合に、主記憶にアクセスが必要になる頻度を低くして、キャッシュミス・ペナルティーを軽減することができる。

【００２９】
制御部１１４は、命令フェッチアドレスと命令フェッチリクエストとを１次命令キャッシュ４０に発行し、この命令フェッチアドレスから命令をフェッチする。制御部１１４は、フェッチした命令をデコードした結果に従い演算部１１５を制御して、フェッチされた命令を実行する。演算制御部１１７は、制御部１１４の制御下で動作し、演算対象のレジスタ１１６からのデータを演算器１１８に供給したり、演算結果のデータを指定されたレジスタ１１６に格納したりする。また演算制御部１１７は、演算器１１８が実行する演算のタイプを指定する。更に演算制御部１１７は、アクセス先のアドレスを指定し、１次キャッシュ部１１３の当該アドレスに対してロード命令やストア命令を実行する。ロード命令により、指定アドレスから読み出されたデータは、指定されたレジスタ１１６に格納される。またストア命令により、指定されたレジスタ１１６のデータが、指定されたアドレスに書き込まれる。
【００３０】
まず、Ｏｒａｃｌｅ－ｎｕｍｂｅｒのサブセットであるｏｒａｃｌｅｎｕｍ６４を定義する。ｏｒａｃｌｅｎｕｍ６４は有効な仮数部の長さが７バイト以下であるようなＯｒａｃｌｅ－ｎｕｍｂｅｒである。
【００３１】
図３は、ｏｒａｃｌｅｎｕｍ６４の構成を示す図である。ｏｒａｃｌｅｎｕｍ６４の数を表現するデータ１２１は、８バイト長のレジスタに格納することができる。なお、符号及び指数部を含めた長さが８バイトに満たないＯｒａｃｌｅ－ｎｕｍｂｅｒをｏｒａｃｌｅｎｕｍ６４としてレジスタに格納する際には、図３にバイト１２２として示すようにデータを左詰めにして格納する。そして右側の余りの部分には、０Ｘ００の値を有するバイト１２３を余りのバイト数に等しい数だけ格納する。
【００３２】
図４は、長さが９バイトのｏｒａｃｌｅｎｕｍ６４の構成を示す図である。図４に示すように、長さが９バイトである負のＯｒａｃｌｅ－ｎｕｍｂｅｒのデータ１２５は、７バイト長の仮数部１２６に続いて最後のバイト（第９バイト）にターミネータ１２７（０Ｘ６６）が付加されている。このＯｒａｃｌｅ－ｎｕｍｂｅｒは、有効な仮数部１２６の長さが７バイトであるので、ｏｒａｃｌｅｎｕｍ６４である。このことはｏｒａｃｌｅｎｕｍ６４を符号反転した数が必ずｏｒａｃｌｅｎｕｍ６４となるために必要である。
【００３３】
図５は、ｏｒａｃｌｅｎｕｍ６４を直接演算対象とできる演算器の構成の一例である。図５の演算器は、図２の演算回路１１９の一部分に相当する。図５に示す演算器は、入力Ｘレジスタ１３１、入力Ｙレジスタ１３２、内部形式変換回路１３３及び１３４、指数仮数演算回路１３５、セレクタ１３６及び１３７、シフタ１３８及び１３９、指数加算器１４０、及び絶対値加算器１４１を含む。演算器は更に、正規化回路１４２、丸め回路１４３、外部形式変換回路１４４、及び出力Ｚレジスタ１４５を含む。内部形式変換回路１３３及び１３４と外部形式変換回路１４４とを工夫することで、例えばｏｒａｃｌｅｎｕｍ６４とＩＥＥＥ７５４－ｄｅｃｉｍａｌ６４との両方に対応させることも可能である。図５において、入力と出力は同形式同精度の浮動小数点数とする。入力データは正規化されていなくてもよい。出力データは正規化される。入力データはｏｒａｃｌｅｎｕｍ６４フォーマットとし、正規化されていないデータも解釈可能である。出力データはｏｒａｃｌｅｎｕｍ６４フォーマットの通りに出力される。
【００３４】
内部形式変換回路１３３及び１３４により、入力を符号部、指数部、及び仮数部に分割し、入力の値表現を内部形式に変換する。入力Ｘの符号、指数、仮数をそれぞれ符号Ｘ、指数Ｘ、仮数Ｘとする。入力Ｙの符号、指数、仮数をそれぞれ符号Ｙ、指数Ｙ、仮数Ｙとする。
【００３５】
指数仮数演算回路１３５が、指数Ｘ及び指数Ｙと、仮数Ｘ及び仮数Ｙとを受け取る。指

数仮数演算回路１３５は、指数Ｘと指数Ｙの大小比較をする。大小比較の結果に基づいて、指数仮数演算回路１３５は、指数の大きい側の仮数（第１の仮数）がシフタ１３８に入力され、且つ、指数の小さい側の仮数（第２の仮数）がシフタ１３９に入力されるように、セレクト信号を生成する。指数仮数演算回路１３５は、指数Ｘと指数Ｙとの差の絶対値と第１の仮数の先行ゼロ計数値とを比較し、指数Ｘと指数Ｙとの差の絶対値の方が大きい場合は、第１の仮数の先行ゼロ計数値をシフタ１３８の左シフト量として出力する。指数Ｘと指数Ｙとの差の絶対値の方が小さい場合は、指数仮数演算回路１３５は、指数Ｘと指数Ｙとの差の絶対値をシフタ１３８の左シフト量として出力する。ここで、先行ゼロ計数値とは、仮数部の最上位桁から連続するゼロを計数した値をいう。

【００３６】

指数仮数演算回路１３５は、指数Ｘと指数Ｙとの差の絶対値と第１の仮数の先行ゼロ計数値とを比較し、指数Ｘと指数Ｙとの差の絶対値の方が大きい場合は、指数Ｘと指数Ｙとの差の絶対値から第１の仮数の先行ゼロ計数値を減算した値をシフタ１３９の右シフト量として出力する。指数Ｘと指数Ｙとの差の絶対値の方が小さい場合は、指数仮数演算回路１３５は、ゼロをシフタ１３９の右シフト量として出力する。指数仮数演算回路１３５は更に、指数の小さい側に上述の右シフト量を加算した値を指数として出力する。

【００３７】

シフタ１３８は、入力されたシフト量に基づき、入力された仮数を左シフトする。シフタ１３９は、入力されたシフト量に基づき、入力された仮数を右シフトする。各シフタのシフト結果は絶対値加算器１４１に入力される。

【００３８】

減算の場合は、片方の仮数が反転され、絶対値加算器１４１にキャリーが入力される。絶対値加算器１４１による加算の結果桁あふれが生じた場合は、１桁右にシフトした値が出力される。同時に、絶対値加算器１４１から指数加算器１４０にキャリーが送られ、送られたキャリーが指数に加算される。

【００３９】

絶対値加算器１４１による加算の結果桁落ちが生じた場合は、１桁左にシフトした値が出力される。同時に、絶対値加算器１４１から指数加算器１４０に桁落ちを示す信号が送られ、指数から減算される。乗算や除算の場合は、加算結果を再利用するループ演算回路を用いてもよい。

【００４０】

正規化回路１４２が加算結果を受け取り、正規化演算結果を出力する。丸め回路１４３が、正規化演算結果を丸める。外部形式変換回路１４４は、丸め処理後の正規化演算結果を外部形式に変換し、出力Ｚレジスタ１４５に出力する。

【００４１】

図６は、指数仮数演算回路の構成の一例を示す図である。図６に示す指数仮数演算回路１３５は、比較回路１５１、絶対値加算器１５２、セレクタ１５３及び１５４、加算器１５５、先行ゼロ計数回路１５６、セレクタ１５７及び１５８、及び加算器１５９を含む。

【００４２】

比較回路１５１は、指数Ｘと指数Ｙとの大小比較をし、指数の大きい側の仮数（第1の仮数）がシフタ１３８に入力され、指数の小さい側の仮数（第２の仮数）がシフタ１３９に入力されるように、セレクト信号を生成する。絶対値加算器１５２は、指数Ｘと指数Ｙの差の絶対値を計算する。先行ゼロ計数回路１５６は、セレクトされた仮数の先行ゼロを計数する。加算器１５５は、指数Ｘと指数Ｙとの差の絶対値とセレクトされた仮数の先行ゼロ計数値とを比較し、比較結果に応じたセレクト信号を出力する。セレクタ１５８は、差の絶対値の方が大きい場合、先行ゼロ計数値をシフタ１３８の左シフト量として出力する。セレクタ１５８は、差の絶対値の方が小さい場合、差の絶対値をシフタ１３８の左シフト量として出力する。

【００４３】

セレクタ１５７は、差の絶対値の方が大きい場合、差の絶対値から先行ゼロ計数値を減

算した値をシフタ１３９の右シフト量として出力する。セレクタ１５７は、差の絶対値の方が小さい場合、ゼロをシフタ１３９の右シフト量として出力する。

【００４４】

加算器１５９は、セレクタ１５３から小さい側の指数を受け取る。加算器１５９は、受け取った小さい側の指数に上述の右シフト量を加算して求めた値を、指数として出力する。

【００４５】

図７は、正規化回路の構成の一例を示す図である。図７に示す正規化回路１４２は、先行ゼロ計数回路１６０、シフト量補正回路１６１、左シフタ１６２、指数演算器１６３、ビットシフタ１６４及び１６５、及びセレクタ１６６を含む。

【００４６】

仮数は桁あふれを考慮する場合は最上位側が１桁多い幅で入力される。先行ゼロ計数回路１６０は、最上位桁の１つ上の桁を除いた仮数を受け取り、先行ゼロを計数することにより得られた計数値を出力する。シフト量補正回路１６１は、先行ゼロ計数値と指数の最下位ビットとを受け取る。先行ゼロ計数値の最下位ビット（偶奇を表す）と指数の最下位ビットとをＸＯＲ（排他的論理和）した値が１の場合、シフト量補正回路１６１は、先行ゼロ計数値から1を減じた値を左シフト量として出力する。また上記ＸＯＲ値が０の場合、シフト量補正回路１６１は、先行ゼロ計数値を左シフト量として出力する。

【００４７】

左シフタ１６２は、最上位桁の１つ上の桁を除いた仮数とシフト量とを受け取り、指定されたシフト量だけ仮数を左シフトした値を出力する。セレクタ１６６は、入力の最上位桁の１つ上の桁が０の場合、左シフタ１６２が出力する左シフトした結果を選択する。またセレクタ１６６は、入力の最上位桁の１つ上の桁が１であり且つ指数の最下位ビットが０の場合、最上位桁の１つ上の桁を含む入力仮数を２桁右シフトした値を選択する。またセレクタ１６６は、入力の最上位桁の１つ上の桁が１であり且つ指数の最下位ビットが１の場合、最上位桁の１つ上の桁を含む入力仮数を1桁右シフトした値を選択する。セレクタ１６６の選択した値は、仮数として出力される。

【００４８】

指数演算器１６３は、指数、シフト量、及び入力仮数の最上位桁を受け取る。入力仮数の最上位桁が０の場合、指数演算器１６３は、指数からシフト量を減算した結果を指数として出力する。入力仮数の最上位桁が１であり且つ指数の最下位ビットが０の場合、指数演算器１６３は、指数に２を加算した結果を指数として出力する。入力仮数の最上位桁が１であり且つ指数の最下位ビットが１の場合、指数演算器１６３は、指数に１を加算した結果を指数として出力する。

【００４９】

図８は、先行ゼロ計数回路の構成の一例を示す図である。図８（ａ）に示すように、先行ゼロ計数回路は、変換回路１６０を含む。変換回路１６０は仮数を入力データとして受け取り、図８（ｂ）に示すテーブルに従い入力データから出力データを生成する。この出力データが先行ゼロ計数値であり、２進数により計数値が表わされる。テーブルでは、一番左側のＸはゼロ以外の値を表し、それ以外のＸはドントケアを表す。０は計数対象となるゼロを表す。

【００５０】

図９は、内部形式変換回路の構成の一例を示す図である。図９に示す内部形式変換回路１３３又は１３４は、セレクタ１７０乃至１７４、加算器１７５及び１７６、及び２進１０進変換回路１７７及び１７８を含む。

【００５１】

入力されたデータは、符号部Ｓ、指数部ＥＸＰ、仮数部Ｍ０１、Ｍ０２、・・・に分割される。符号部Ｓは1ビットであり、符号としてそのまま出力される。符号は１の場合に正を表す。

【００５２】

指数部ＥＸＰは７ビットである。セレクタ１７０は、符号が１の場合、指数部ＥＸＰをそのまま指数として出力する。またセレクタ１７０は、符号が０の場合、指数部ＥＸＰを反転した値を指数として出力する。

【００５３】

仮数部Ｍ０１、Ｍ０２、・・・の各々は、８ビット長のデータである。セレクタ１７１は、符号が１の場合は仮数を選択し、符号が０の場合は仮数の反転値を選択する。この選択値が加算器１７５の一方の入力となる。またセレクタ１７２は、符号が１の場合は－１を選択し、符号が０の場合は＋１０１を選択する。この選択値が加算器１７５のもう一方の入力となる。また、符号が０の場合には、加算器１７５にキャリーが入力される。セレクタ１７３及び１７４並びに加算器１７６についても同様である。

【００５４】

加算器１７５及び１７６の出力が、それぞれ、２進１０進変換回路１７７及び１７８により２進形式からＢＣＤ形式の値に変換され、ＢＣＤ変換後の値が仮数として出力される。ただし、加算器からのＣＯが０の場合には、変換後のＢＣＤ値は強制的にゼロとなる。

【００５５】

図１０は、外部形式変換回路の構成の一例を示す図である。図１０に示す外部形式変換回路１４４は、デコーダ１８０、１０進２進変換回路１８１及び１８２、セレクタ１８３乃至１８７、及び加算器１８８及び１８９を含む。図１０では、２つの仮数部ＢＣＤ０１、ＢＣＤ０２のみに対する回路部分が明示的に示されるが、仮数部の数が２以上の場合には、同様の回路部分がそれらの仮数部に対して設けられる。

【００５６】

入力された１ビットの符号部は、そのまま符号として出力される。符号は１の場合に正を表す。指数部の入出力は７ビットである。セレクタ値１８３により、符号が１の場合は入力指数がそのまま出力指数となり、符号が０の場合は入力指数の反転値が出力指数となる。仮数部ＢＣＤ０１、ＢＣＤ０２、・・・は、１つの入力ＢＣＤあたり８ビットであり、１０進２進変換回路１８１、１８２、・・・によりＢＣＤから２進数に変換される。

【００５７】

ターミネータ選択信号を受け取るデコーダ１８０は、終端の桁を現すターミネータ桁を選択するターミネータ桁セレクト信号を生成する。このターミネータ桁セレクト信号は、後段のセレクタ１８４、１８５、・・・に分配される。

【００５８】

加算器１８８、１８９、・・・の各々において、一方の入力には、符号が１の場合は仮数がそのまま入力され、符号が０の場合は仮数の反転値が入力される。但し、ターミネータ桁セレクト信号が１の場合は、ゼロが選択されて入力される。もう一方の入力には、符号が１の場合は＋１が入力され、符号が０の場合は＋１０１が入力される。但し、ターミネータ桁セレクト信号が１の場合は、＋１０１が選択されて入力される。ここで実際に加算したい値は＋１０２であるが、＋１０１にキャリーインが加算されることで、＋１０２を加算することと同等になる。また、符号が０の場合には加算器にキャリーインが入力される。加算器１８８、１８９、・・・の出力が、各桁の仮数として出力される。

【００５９】

図１１は、固定精度浮動小数点数加算の具体例を示す図である。図１１において、入力Ｘの指数をＥｘ、入力Ｙの指数をＥｙ、入力Ｘの先行ゼロ計数値をＬｘ、入力Ｙの先行ゼロ計数値をＬｙとする。また出力Ｚの指数をＥｚとする。可能な限り高い演算精度を保持するとき、指数値の大きい側を左シフトして桁合わせを行う。但し、先行ゼロの桁数より多く左シフトすることは、上位桁が欠落してしまうため、桁合わせに必要な左シフト量Ｅｘ－Ｅｙが先行ゼロの桁数を上回る場合は、指数値の小さい側を右シフトすることで、桁合わせを行う。このためにＥｘ－Ｅｙを計算し、Ｌｘと比較する。図１１に示す具体例ではＬｘのほうが大きい。即ち、左シフト量Ｅｘ－Ｅｙが先行ゼロの数内に収まるので、入力Ｘの仮数１９１のみ左シフトされる。入力Ｙの仮数１９２は右シフトされない。左シフト量はＥｘ－Ｅｙであり、右シフト量は０である。

【００６０】
このようにして桁合わせされた数同士、即ち仮数１９１を左シフトして得られた仮数１９３と仮数１９２とを加算する。更に、加算結果１９４を正規化し、先行ゼロ計数値が１以上の場合はその分だけ左シフトを行い、桁あふれがある場合には１桁分の右シフトを行う。但し、左シフトした結果指数が奇数になる場合は、左シフト量を１減らす。または右シフトした結果指数が奇数になる場合は、右シフト量を１増やす。指数は、左右シフト量に合わせて調整する。左シフトした場合は左シフト量を指数から減算し、右シフトした場合は右シフト量を指数に加算する。本具体例では１桁左シフトが必要だが、指数が０から１となり奇数になるので、左シフト量は１減らされて０となり、指数は０のままである。その結果、正規化により加算結果は変化せず、そのまま演算結果の仮数１９４及び対応する指数Ｅｚを含む演算結果として出力される。

【００６１】
図１２は、固定精度浮動小数点数加算の別の具体例を示す図である。この具体例では必要な左シフト量Ｅｘ－ＥｙよりもＬｘのほうが小さい。この場合、必要な左シフト量Ｅｘ－Ｅｙが先行ゼロ計数値の数内に収まらないので、入力Ｘの仮数２０１をＬｘ分だけ左シフトし、左シフトしきれなかった分だけ入力Ｙの仮数２０２を右シフトする。即ち、左シフト量はＬｘであり、右シフト量は（Ｅｘ－Ｅｙ）－Ｌｘである。

【００６２】
このようにして桁合わせされた数同士、即ち仮数２０１を左シフトして得られた仮数２０３と仮数２０２を右シフトして得られた仮数２０４とを加算する。このとき、右シフトによって溢れた桁も保持する。更に、加算結果２０５を正規化し、先行ゼロ計数値が１以上の場合はその分だけ左シフトを行い、桁あふれがある場合には１桁分の右シフトを行う。但し、左シフトした結果指数が奇数になる場合は、左シフト量を１減らす。または右シフトした結果指数が奇数になる場合は、右シフト量を１増やす。指数は、左右シフト量に合わせて調整する。左シフトした場合は左シフト量を指数から減算し、右シフトした場合は右シフト量を指数に加算する。本具体例では１桁右シフトが必要だが、指数Ｅｚが２から３となり奇数になるので、右シフト量は１増やされて２となり、指数Ｅｚは２から４となる。

【００６３】
その後、正規化した結果２０６に対して丸め処理を実行する。この例においては右にはみ出した桁を四捨五入することにより実行される。このようにして丸められた結果が、丸め結果の仮数２０７及び対応する指数Ｅｚを含む演算結果として出力される。

【００６４】
以下に、１つのＯｒａｃｌｅ－ｎｕｍｂｅｒを複数のｏｒａｃｌｅｎｕｍ６４を含む組により表現する方法について説明する。

【００６５】
最大で２１バイト長のＯｒａｃｌｅ－ｎｕｍｂｅｒの計算を、ｏｒａｃｌｅｎｕｍ６４演算ハードウエアを用いて実現するためには、Ｏｒａｃｌｅ－ｎｕｍｂｅｒを複数のｏｒａｃｌｅｎｕｍ６４を含む組で表現する必要がある。

【００６６】
図１３は、２１バイト長のＯｒａｃｌｅ－ｎｕｍｂｅｒを３つの部分に分ける方法を示す図である。図２２に示されるように、最大２０バイト長の仮数部２１０は、７バイトの仮数部２１１、７バイトの仮数部２１２、及び６バイトの仮数部２１３に分割される。

【００６７】
図１４は、分割された３つの仮数部にそれぞれ対応するｏｒａｃｌｅｎｕｍ６４数を生成する方法を示す図である。ａ０については、元のＯｒａｃｌｅ－ｎｕｍｂｅｒの先頭８バイト部分（１バイトの符号及び指数部２１４＋７バイトの仮数部２１１）を切り出してくるだけで、ｏｒａｃｌｅｎｕｍ６４形式の数が得られる。ａ１及びａ２については、元のＯｒａｃｌｅ－ｎｕｍｂｅｒの第１バイト（符号及び指数部２１４）を加工する必要がある。具体的には、元のＯｒａｃｌｅ－ｎｕｍｂｅｒの基数１０での指数をＥとして、ａ

１の指数Ｅ１＝Ｅ－１４とすることで、ａ１についての１バイトの符号及び指数部２１５を生成する。この符号及び指数部２１５に７バイトの仮数部２１２を付加して、ａ１に相当するｏｒａｃｌｅｎｕｍ６４形式の数が得られる。またａ２の指数Ｅ２＝Ｅ－２８とすることで、ａ２についての１バイトの符号及び指数部２１６を生成する。この符号及び指数部２１６に６バイトの仮数部２１３及び１バイトの"０"を付加して、ａ２に相当するｏｒａｃｌｅｎｕｍ６４形式の数が得られる。このようにして得られた３つのｏｒａｃｌｅｎｕｍ６４の組を、以下においてｔｒｉｐｌｅ－ｏｒａｃｌｅｎｕｍ６４と呼ぶ。

【００６８】

以下に、ｔｒｉｐｌｅ－ｏｒａｃｌｅｎｕｍ６４形式に対する四則演算をｏｒａｃｌｅｎｕｍ６４演算器により実行する構成について説明する。まず、最も基本となる、ｏｒａｃｌｅｎｕｍ６４の数同士の正確な和を求める演算について説明する。以下に示すｔｗｏ－ｓｕｍ（２数の和）は、非特許文献１にｆｏｒｍｕｌａ（４．１６）として示されているものであり、同様のものが非特許文献２のｐ．１８、及び、非特許文献３のａｌｇｏｒｉｔｈｍ４に示されている。

【００６９】

```
Two_sum(X, Y)
  z  = fl(X + Y)
  w  = fl(z - X)
  v  = fl(z - w)
  z1 = fl(Y - w)
  z2 = fl(v - X)
  zz = fl(z1 - z2)
  return(z, zz)
```

ここでｆｌ（Ｘ＋Ｙ）は、Ｘ＋Ｙの真の値を、浮動小数点数にマッピングした結果、即ち浮動小数点形式の限られた精度内で表現した結果を示す。上記のＴｗｏ＿ｓｕｍによって得られる２数ｚ及びｚｚは、正確な意味においてｚ＋ｚｚ＝Ｘ＋Ｙを満たす。ｚはＸ＋Ｙの最重要部（most significant part）を固定精度浮動小数点の精度内で表現した値であり、ｚｚは固定精度浮動小数点の精度で表現しきれなかった残余分を表す。

【００７０】

マッピング時に発生する丸め処理方法として、１０進数で代表的な丸め方法である四捨五入を例に取り説明する。簡単のために固定精度浮動小数点数の精度は１０進２桁とする。この場合、２００００と－１との２数の和は、上記のＴｗｏ＿ｓｕｍにより以下のように計算される。

【００７１】

```
X = 20000
Y =    -1
z = fl(X＋Y) = 20000
w = fl(z－X) = 0
v = fl(z－w) = 20000
z1 = fl(Y－w) = -1
z2 = fl(v－X) = 0
zz = fl(z1－z2) = -1
```

この例のように、一般に、ｔｗｏ＿ｓｕｍの結果であるｚとｚｚは異符合となり得る。一方、丸め方法を切り捨てとした場合は、ＸとＹとで有効桁がオーバーラップしない場合には、以下の例のようにｚ＋ｚｚがＸ＋Ｙと等しくならない場合がある。

【００７２】

```
X = 20000
Y =    -1
z = fl(X＋Y) = 19000
w = fl(z－X) = -1000
```

v = fl(z－w) = 20000
z1 = fl(Y－w) = 990
z2 = fl(v－X) = 0
zz = fl(z1－z2) = 990

ここで、Xの絶対値がYの絶対値より大きいか等しいと仮定し、更に以下のような新たな丸め処理を行うことにより、zzの算出が容易になることを示す。

【００７３】

（１）XとYとの間で有効桁のオーバーラップが無い場合は四捨五入する
（２）XとYとの間で有効桁のオーバーラップがある場合は切り捨てる

但し、XとYとの間で有効桁がオーバーラップしていない場合であっても、XとYとの間で有効桁が連続している場合は（２）に含めることにする。

【００７４】

具体的な例を用いて、上記のような丸め処理を適用することの効果を説明する。前述の場合と同様に固定精度浮動小数点数の精度は１０進２桁である。また、XとYとの間で有効桁が連続している場合の例を示す。まず上記の特殊な丸め処理ではなく、flが常に四捨五入の丸め処理を行なう場合には、以下のようになる。

【００７５】

X = 2000
Y = 52
z = 2100
w = 100
v = 2000
z1 = -48
z2 = 0
zz = -48

それに対して、上記の新たな丸め処理を用いる場合は、以下のようになる。

【００７６】

X = 2000
Y = 52
z = 2000
w = 0
v = 2000
z1 = 52
z2 = 0
zz = 52

上記の新たな丸め処理を用いることにより、以下の効果が得られる。まず（１）の場合には、オーバーラップがないので、X＋Yを行った後の四捨五入による丸め処理や、四捨五入によって発生したwを補正するために必要であったz1を求める演算処理がなくなる。また（２）の場合には、X＋Yの精度は有効桁数の２倍以下であることが保証され、丸め処理が切り捨てのみとなるため、演算を実行するハードウエアの実現が容易となる。

【００７７】

以上のことから、２つの入力値の絶対値を比較して、その比較結果に応じて上記の（１）又は（２）の場合分けを行ない、新たな丸め処理を実行すれば、zzの算出が容易となる。また更に、z，zzを求める演算get_z（x，y），get_zz（x，y）をハードウエアで実行する回路を設ければ、以下に示すように、two_sumを高速に処理することが可能となる。

【００７８】

```
Two_sum_fast(x, y)
  z = get_z(x, y)
  zz = get_zz(x, y)
```

return(z, zz)

図１５は、ｇｅｔ＿ｚ演算を実現する回路の構成の一例を示す図である。図１５の演算器は、図２の演算回路１１９の一部分に相当する。図１５に示す演算器は、入力Ｘレジスタ２２１、入力Ｙレジスタ２２２、内部形式変換回路２２３及び２２４、指数仮数演算回路２２５、セレクタ２２６及び２２７、シフタ２２８及び２２９、指数加算器２３０、及び絶対値加算器２３１を含む。演算器は更に、セレクタ２３２及び２３３、正規化回路２３４、外部形式変換回路２３５、及び出力Ｚレジスタ２３６を含む。図１５において、図５に示す回路と同一又は対応する構成要素は同一又は対応する名称で参照する。図１５において、入力と出力は同形式同精度の浮動小数点数とする。入力データは正規化されていなくてもよい。出力データは正規化される。入力データはｏｒａｃｌｅｎｕｍ６４フォーマットとし、正規化されていないデータも解釈可能である。出力データはｏｒａｃｌｅｎｕｍ６４フォーマットの通りに出力される。

【００７９】

内部形式変換回路２２３及び２２４により、入力を符号部、指数部、及び仮数部に分割し、入力の値表現を内部形式に変換する。入力Ｘの符号、指数、仮数をそれぞれ符号Ｘ、指数Ｘ、仮数Ｘとする。入力Ｙの符号、指数、仮数をそれぞれ符号Ｙ、指数Ｙ、仮数Ｙとする。

【００８０】

指数仮数演算回路２２５が、指数Ｘ及び指数Ｙと、仮数Ｘ及び仮数Ｙとを受け取る。指数仮数演算回路２２５は、指数Ｘと指数Ｙの大小比較をする。大小比較の結果に基づいて、指数仮数演算回路２２５は、指数の大きい側の仮数（第１の仮数）がシフタ２２８に入力され、且つ、指数の小さい側の仮数（第２の仮数）がシフタ２２９に入力されるように、セレクト信号を生成する。指数仮数演算回路２２５は、指数Ｘと指数Ｙとの差の絶対値と第１の仮数の先行ゼロ計数値とを比較し、前者の方が大きい場合は、後者をシフタ２２８の左シフト量として出力する。前者の方が小さい場合は、指数仮数演算回路２２５は、前者をシフタ２２８の左シフト量として出力する。

【００８１】

指数仮数演算回路２２５は、指数Ｘと指数Ｙとの差の絶対値と第１の仮数の先行ゼロ計数値とを比較し、指数Ｘと指数Ｙとの差の絶対値の方が大きい場合は、指数Ｘと指数Ｙとの差の絶対値から第１の仮数の先行ゼロ計数値を減算した値をシフタ２２９の右シフト量として出力する。指数Ｘと指数Ｙとの差の絶対値の方が小さい場合は、指数仮数演算回路２２５は、ゼロをシフタ２２９の右シフト量として出力する。指数仮数演算回路２２５は更に、指数の小さい側に上述の右シフト量を加算した値を指数として出力する。

【００８２】

指数仮数演算回路２２５は更に、（指数Ｘ－仮数Ｘの先行ゼロ計数値）－（指数Ｙ－仮数Ｙの先行ゼロ計数値）の絶対値が１４以上であるか否かを判定する。この絶対値が１４以上である場合、指数仮数演算回路２２５は、第１の仮数と第１の仮数に対応する指数とがセレクタ２３２及び２３３により選択されるように、セレクト信号を生成する。これにより、上記絶対値が１４以上である場合には、第１の仮数が、シフタ２２８及び絶対値加算器２３１をバイパスして、正規化回路２３４に入力されることになる。

【００８３】

シフタ２２８は、入力されたシフト量に基づき、入力された仮数を左シフトする。シフタ２２９は、入力されたシフト量に基づき、入力された仮数を右シフトする。各シフタのシフト結果は絶対値加算器２３１に入力される。

【００８４】

減算の場合は、片方の仮数が反転され、絶対値加算器２３１にキャリーが入力される。絶対値加算器２３１による加算の結果桁あふれが生じた場合は、１桁右にシフトした値が出力される。同時に、絶対値加算器２３１から指数加算器２３０にキャリーアウトが送られ、指数に加算される。

【００８５】

絶対値加算器２３１による加算の結果桁落ちが生じた場合は、１桁左にシフトした値が出力される。同時に、絶対値加算器２３１から指数加算器２３０に桁落ちを示す信号が送られ、指数から減算される。

【００８６】

セレクタ２３２及び２３３は、指数仮数演算回路２２５によって生成されたセレクト信号に応じて、加算結果の指数及び仮数、又は、第１の仮数及びそれに対応する指数を選択する。正規化回路２３４が、セレクタ２３２及び２３３により選択された指数及び仮数を受け取り、正規化演算結果を出力する。外部形式変換回路２３５は、正規化演算結果を外部形式に変換し、出力Ｚレジスタ２３６に出力する。

【００８７】

図１６は、ｇｅｔ＿ｚ演算の具体例を示す図である。図１６において、入力Ｘの指数をＥｘ、入力Ｙの指数をＥｙ、入力Ｘの先行ゼロ計数値をＬｘ、出力Ｚの指数をＥｚとする。この例において実行する演算は、図１６に示すｇｅｔ＿ｚ演算２４０である。可能な限り高い演算精度を保持するとき、指数値の大きい側を左シフトして桁合わせを行う。但し、先行ゼロの桁数より多く左シフトすることは、上位桁が欠落してしまうために不可能であるので、桁合わせに必要な左シフト量Ｅｘ－Ｅｙが先行ゼロの桁数を上回る場合は、指数値の小さい側を右シフトすることで、桁合わせを行う。このためにEｘ－Eｙを計算し、Ｌｘと比較する。図１６に示す具体例ではLｘのほうが大きい。即ち、左シフト量Ｅｘ－Ｅｙが先行ゼロの数内に収まるので、入力Ｘの仮数２４１のみ左シフトされる。入力Ｙの仮数２４２は右シフトされない。左シフト量はEｘ－Eｙであり、右シフト量は０である。

【００８８】

このようにして桁合わせされた数同士、即ち仮数２４１を左シフトして得られた仮数２４３と仮数２４２のままである仮数２４４とを加算する。更に、加算結果２４５を正規化し、先行ゼロ計数値が１以上の場合は先行ゼロ計数値だけ左シフトを行い、桁あふれがある場合には１桁分の右シフトを行う。但し、左シフトした結果指数が奇数になる場合は、左シフト量を１減らす。または右シフトした結果指数が奇数になる場合は、右シフト量を１増やす。指数は、左右シフト量に合わせて調整する。左シフトした場合は左シフト量を指数から減算し、右シフトした場合は右シフト量を指数に加算する。本具体例では１桁左シフトが必要だが、指数が０から１となり奇数になるので、左シフト量は１減らされて０となり、指数は０のままである。この例では、正規化による変化はない。更に正規化結果の仮数２４６の上位桁を選択する。例えば入力フォーマットの仮数が１４桁である場合は、最上位から１４桁を上位桁、最上位から１５桁目以降を下位桁とする。

【００８９】

上記の演算の結果、仮数２４７及び対応する指数Ｅｚを含む演算結果が出力される。

【００９０】

図１７は、ｇｅｔ＿ｚ演算の別の具体例を示す図である。この例において実行する演算は、図１７に示すｇｅｔ＿ｚ演算２５０である。可能な限り高い演算精度を保持するとき、指数値の大きい側を左シフトして桁合わせを行う。但し、先行ゼロの桁数より多く左シフトすることは、上位桁が欠落してしまうために不可能であるので、桁合わせに必要な左シフト量Ｅｘ－Ｅｙが先行ゼロの桁数を上回る場合は、指数値の小さい側を右シフトすることで、桁合わせを行う。このためにEｘ－Eｙを計算し、Ｌｘと比較する。図１７に示す具体例ではLｘのほうが小さい。即ち、左シフト量Ｅｘ－Ｅｙが先行ゼロの数内に収まらないので、入力Ｘの仮数２５１をＬｘ分だけ左シフトし、左シフトしきれなかった分だけ入力Ｙの仮数２５２を右シフトする。即ち、左シフト量はＬｘであり、右シフト量は（Ｅｘ－Ｅｙ）－Ｌｘである。

【００９１】

このようにして桁合わせされた数同士、即ち仮数２５１を左シフトして得られた仮数２５３と仮数２５２を右シフトして得られた仮数２５４とを加算する。このとき、右シフトによって溢れた桁も保持する。更に、加算結果２５５を正規化し、先行ゼロが１以上の場合はその分だけ左シフトを行い、桁あふれがある場合には１桁分の右シフトを行う。但し

、左シフトした結果指数が奇数になる場合は、左シフト量を１減らす。または右シフトした結果指数が奇数になる場合は、右シフト量を１増やす。指数は、左右シフト量に合わせて調整する。左シフトした場合は左シフト量を指数から減算し、右シフトした場合は右シフト量を指数に加算する。本具体例では１桁右シフトが必要だが、指数が２から３となり奇数になるので、右シフト量は１増やされて２となり、指数は２から４となる。このようにして得られた正規化の結果の仮数２５６の上位桁を選択する。例えば入力フォーマットの仮数が１４桁である場合は、最上位から１４桁を上位桁、最上位から１５桁目以降を下位桁とする。

【００９２】

上記の演算の結果、演算結果の仮数２５７が出力されるとともに、対応する指数Ｅｚが演算結果の指数として出力される。

【００９３】

図１８は、ｇｅｔ＿ｚ演算の更に別の具体例を示す図である。この例において実行する演算は、図１８に示すｇｅｔ＿ｚ演算２６０である。可能な限り高い演算精度を保持するとき、指数値の大きい側を左シフトして桁合わせを行う。但し、先行ゼロの桁数より多く左シフトすることは、上位桁が欠落してしまうため、桁合わせに必要な左シフト量Ｅｘ－Ｅｙが先行ゼロの桁数を上回る場合は、指数値の小さい側を右シフトすることで、桁合わせを行う。但し、桁合わせした結果、２つの入力の間に１４桁以上の差がある場合、即ち、重なりが全くない場合には、実際に演算処理を実行しなくとも演算結果を求めることが可能である。ｇｅｔ＿ｚは演算結果の上位側を求める演算であり、２入力の重なりがない場合には値の大きい側がそのまま上位側となるからである。

【００９４】

この条件を満たすか否かを確認するため、（Ｅｘ－Ｌｘ）－（Ｅｙ－Ｌｙ）を計算し、計算結果が１４以上の場合には上記の場合に該当すると判定する。ここで値１４は、使用フォーマットにおける仮数の桁数である。この具体例では上記の場合に該当し、仮数２６１と仮数２６２との間に重なりが存在しない。この場合、仮数２６１がそのままバイパスされて加算結果として取り扱われる。

【００９５】

更にバイパスされた結果２６３を正規化し、先行ゼロ計数値が１以上の場合は先行ゼロ計数値だけ左シフトを行い、桁あふれがある場合には１桁分の右シフトを行う。但し、左シフトした結果指数が奇数になる場合は、左シフト量を１減らす。または右シフトした結果指数が奇数になる場合は、右シフト量を１増やす。指数は、左右シフト量に合わせて調整する。左シフトした場合は左シフト量を指数から減算し、右シフトした場合は右シフト量を指数に加算する。本具体例では３桁左シフトが必要だが、指数が１６から１３となり奇数になるので、左シフト量は１減らされて２となり、指数は１６から１４となる。このようにして得られた正規化の結果が、そのまま演算結果の仮数２６４及び対応する指数Ｅｚを含む演算結果として出力される。

【００９６】

なおｚｚは固定精度の範囲で表現しきれなかった加算演算の残余の部分を表すが、例えば乗算の場合にはｚｚを出力する命令は一般的である。乗算の場合、ｚとｚｚは互いの桁が連続した値である。しかし加算の場合は、２つの入力値の絶対値がフォーマットの桁数以上離れている場合は、ｚとｚｚは互いの桁が連続した値とならない。図１９（ａ）に、２つの入力値の絶対値がフォーマットの桁数以上離れている場合の加算演算の例を示す。この第１のケースにおいては、通常のフローで演算を行いｚｚを出力してもよいが、値の重なりが全くないことを利用すると、演算をしなくとも絶対値の小さい側をそのままｚｚとすることができ、ｚｚを容易に出力することが可能である。

【００９７】

また乗算の場合は、入力の符号は演算結果の仮数の値には影響しないため、ｚｚを求める際に符号を考慮する必要がない。しかし加算の場合は、上記のように２つの入力値の絶対値が離れているとき、互いに異なる符号を有する２つの数の加算（同符号の数同士の減

算）において特別な処理が必要である。図１９（ｂ）に、２つの入力値の絶対値が離れている場合における互いに異なる符号を有する２つの数の加算演算の例を示す。この第２のケースにおいては、通常のフローで演算を行うと、２つの入力の間のゼロで埋められていた桁に、図１９（ｂ）にＡとして示すように、桁下がりによってゼロ以外の値が発生する。その結果、ｚｚを表すために必要な精度が、離れた桁数分だけ増えてしまうことになる。従って、この第２のケースにおいては、通常のフローで演算を実行せずに、上記のケース１と同様の方法で絶対値の小さい側をそのままｚｚとして出力する必要がある。

【００９８】

このように、加算におけるｚｚ出力は、乗算の場合には考慮する必要のないケースを考慮しなければならない。従って、実際の回路構成を考える際には、例えばバイパス回路を追加する等の変更が必要となる点が、乗算におけるｚｚ出力回路と異なる部分となる。

【００９９】

図２０は、ｇｅｔ＿ｚｚ演算を実現する回路の構成の一例を示す図である。図２０の演算器は、図２の演算回路１１９の一部分に相当する。図２０において、図１５と同一又は対応する構成要素は同一又は対応する番号で参照する。図２０に示す演算器は、入力Ｘレジスタ２２１、入力Ｙレジスタ２２２、内部形式変換回路２２３及び２２４、指数仮数演算回路２２５Ａ、セレクタ２２６及び２２７、シフタ２２８及び２２９、指数加算器２３０、及び絶対値加算器２３１を含む。演算器は更に、セレクタ２３２及び２３３、正規化回路２３４、外部形式変換回路２３５、出力Ｚレジスタ２３６、及び正規化回路２７０を含む。

【０１００】

内部形式変換回路２２３及び２２４により、入力を符号部、指数部、及び仮数部に分割し、入力の値表現を内部形式に変換する。入力Ｘの符号、指数、仮数をそれぞれ符号Ｘ、指数Ｘ、仮数Ｘとする。入力Ｙの符号、指数、仮数をそれぞれ符号Ｙ、指数Ｙ、仮数Ｙとする。

【０１０１】

指数仮数演算回路２２５Ａが、指数Ｘ及び指数Ｙと、仮数Ｘ及び仮数Ｙとを受け取る。指数仮数演算回路２２５Ａは、指数Ｘと指数Ｙの大小比較をする。大小比較の結果に基づいて、指数仮数演算回路２２５Ａは、指数の大きい側の仮数（第１の仮数）がシフタ２２８に入力され、且つ、指数の小さい側の仮数（第２の仮数）がシフタ２２９に入力されるように、セレクト信号を生成する。指数仮数演算回路２２５Ａは、指数Ｘと指数Ｙとの差の絶対値と第１の仮数の先行ゼロ計数値とを比較し、指数Ｘと指数Ｙとの差の絶対値の方が大きい場合は、第１の仮数の先行ゼロ計数値をシフタ２２８の左シフト量として出力する。指数Ｘと指数Ｙとの差の絶対値の方が小さい場合は、指数仮数演算回路２２５Ａは、指数Ｘと指数Ｙとの差の絶対値をシフタ２２８の左シフト量として出力する。

【０１０２】

指数仮数演算回路２２５Ａは、指数Ｘと指数Ｙとの差の絶対値と第１の仮数の先行ゼロ計数値とを比較し指数Ｘと指数Ｙとの差の絶対値の方が大きい場合は、指数Ｘと指数Ｙとの差の絶対値から第１の仮数の先行ゼロ計数値を減算した値をシフタ２２９の右シフト量として出力する。第１の仮数の先行ゼロ計数値の方が小さい場合は、指数仮数演算回路２２５Ａは、ゼロをシフタ２２９の右シフト量として出力する。指数仮数演算回路２２５Ａは更に、指数の小さい側に上述の右シフト量を加算した値を指数として出力する。

【０１０３】

指数仮数演算回路２２５Ａは更に、（指数Ｘ－仮数Ｘの先行ゼロ計数値）－（指数Ｙ－仮数Ｙの先行ゼロ計数値）の絶対値が１４以上であるか否かを判定する。この絶対値が１４以上である場合、指数仮数演算回路２２５は、第２の仮数とそれに対応する指数がセレクタ２３２及び２３３により選択されるように、セレクト信号を生成する。これにより、上記（指数Ｘ－仮数Ｘの先行ゼロ計数値）－（指数Ｙ－仮数Ｙの先行ゼロ計数値）の絶対値がフォーマットの桁数である１４以上である場合には、第２の仮数が、シフタ２２８及び絶対値加算器２３１をバイパスして、正規化回路２３４に入力されることになる。

【０１０４】
シフタ２２８は、入力されたシフト量に基づき、入力された仮数を左シフトする。シフタ２２９は、入力されたシフト量に基づき、入力された仮数を右シフトする。各シフタのシフト結果は絶対値加算器２３１に入力される。
【０１０５】
減算の場合は、片方の仮数が反転され、絶対値加算器２３１にキャリーが入力される。絶対値加算器２３１による加算の結果桁あふれが生じた場合は、１桁右にシフトした値が出力される。同時に、絶対値加算器２３１から指数加算器２３０にキャリーが送られ、指数に加算される。
【０１０６】
絶対値加算器２３１による加算の結果桁落ちが生じた場合は、１桁左にシフトした値が出力される。同時に、絶対値加算器２３１から指数加算器２３０に桁落ちを示す信号が送られ、指数から減算される。
【０１０７】
正規化回路２７０が、加算結果と指数演算結果とを受け取り、正規化された指数部と仮数部とを出力する。
【０１０８】
セレクタ２３２及び２３３は、指数仮数演算回路２２５によって生成されたセレクト信号に応じて、正規化された指数及び仮数、又は、第２の仮数及びそれに対応する指数を選択する。正規化回路２３４が、セレクタ２３２及び２３３により選択された指数及び仮数を受け取り、正規化演算結果を出力する。外部形式変換回路２３５は、正規化演算結果を外部形式に変換し、出力Ｚレジスタ２３６に出力する。
【０１０９】
図２１は、ｇｅｔ＿ｚｚ演算の具体例を示す図である。図２１において、入力Ｘの指数をＥｘ、入力Ｙの指数をＥｙ、入力Ｘの先行ゼロ計数値をＬｘ、出力Ｚの指数をＥｚとする。この例において実行する演算は、図２１に示すｇｅｔ＿ｚｚ演算２８０である。可能な限り高い演算精度を保持するとき、指数値の大きい側を左シフトして桁合わせを行う。但し、先行ゼロの桁数より多く左シフトすることは、上位桁が欠落してしまうため、桁合わせに必要な左シフト量Ｅｘ－Ｅｙが先行ゼロの桁数を上回る場合は、指数値の小さい側を右シフトすることで、桁合わせを行う。このためにＥｘ－Ｅｙを計算し、Ｌｘと比較する。図２１に示す具体例ではＬｘのほうが大きい。即ち、左シフト量Ｅｘ－Ｅｙが先行ゼロの数内に収まるので、入力Ｘの仮数２８１のみ左シフトされる。入力Ｙの仮数２８２は右シフトされない。左シフト量はＥｘ－Ｅｙであり、右シフト量は０である。
【０１１０】
このようにして桁合わせされた数同士、即ち仮数２８１を左シフトして得られた仮数２８３と仮数２８２のままである仮数２８４とを加算する。更に、加算結果２８５を正規化し、先行ゼロ計数値が１以上の場合はその分だけ左シフトを行い、桁あふれがある場合には１桁分の右シフトを行う。但し、左シフトした結果指数が奇数になる場合は、左シフト量を１減らす。または右シフトした結果指数が奇数になる場合は、右シフト量を１増やす。指数は、左右シフト量に合わせて調整する。左シフトした場合は左シフト量を指数から減算し、右シフトした場合は右シフト量を指数に加算する。本具体例では１桁左シフトが必要だが、指数が０から１となり奇数になるので、左シフト量は１減らされて０となり、指数は０のままである。
【０１１１】
次に、正規化結果の下位桁２８６を選択する。例えば入力フォーマットの仮数が１４桁である場合は、最上位から１４桁を上位桁、最上位から１５桁目以降を下位桁とする。下位桁選択に伴い、指数は－１４される。本具体例では、下位桁はゼロが選択される。
【０１１２】
更に、選択された下位桁２８７を正規化する。本具体例では対象データはゼロであるので、正規化前の仮数及び指数がそのまま正規化の結果として出力される。正規化された結

果が、演算結果の仮数２８８及び対応する指数Ｅｚを含む演算結果として出力される。

【０１１３】

図２２は、ｇｅｔ＿ｚｚ演算の別の具体例を示す図である。この例において実行する演算は、図２２に示す加算２９０である。可能な限り高い演算精度を保持するとき、指数値の大きい側を左シフトして桁合わせを行う。但し、先行ゼロの桁数より多く左シフトすることは、上位桁が欠落してしまうために不可能であるので、桁合わせに必要な左シフト量Ｅｘ－Ｅｙが先行ゼロの桁数を上回る場合は、指数値の小さい側を右シフトすることで、桁合わせを行う。このためにEx－Eyを計算し、Lxと比較する。図２２に示す具体例ではLxのほうが小さい。即ち、左シフト量Ｅｘ－Ｅｙが先行ゼロの数内に収まらないので、入力Ｘの仮数２９１をＬｘ分だけ左シフトし、左シフトしきれなかった分だけ入力Ｙの仮数２９２を右シフトする。即ち、左シフト量はＬｘであり、右シフト量は（Ｅｘ－Ｅｙ）－Ｌｘである。

【０１１４】

このようにして桁合わせされた数同士、即ち仮数２９１を左シフトして得られた仮数２９３と仮数２９２を右シフトして得られた仮数２９４とを加算する。このとき、右シフトによって溢れた桁も保持する。

【０１１５】

更に、加算結果２９５を正規化し、先行ゼロ計数値が１以上の場合はその分だけ左シフトを行い、桁あふれがある場合には１桁分の右シフトを行う。但し、左シフトした結果指数が奇数になる場合は、左シフト量を１減らす。または右シフトした結果指数が奇数になる場合は、右シフト量を１増やす。指数は、左右シフト量に合わせて調整する。左シフトした場合は左シフト量を指数から減算し、右シフトした場合は右シフト量を指数に加算する。本具体例では１桁右シフトが必要だが、指数が２から３となり奇数になるので、右シフト量は１増やされて４となり、指数は２から４となる。

【０１１６】

次に、正規化結果の下位桁２９６を選択する。例えば入力フォーマットの仮数が１４桁である場合は、最上位から１４桁を上位桁、最上位から１５桁目以降を下位桁とする。下位桁選択に伴い、指数は－１４される。

【０１１７】

更に、選択された下位桁２９７を正規化する。本具体例では、仮数は２桁左にシフトされ、指数は－１０から２が減算されて－１２となる。正規化された結果が、演算結果の仮数２９８及び対応する指数Ｅｚを含む演算結果として出力される。

【０１１８】

図２３は、ｇｅｔ＿ｚｚ演算の更に別の具体例を示す図である。この例において実行する演算は、図２３に示すｇｅｔ＿ｚｚ演算３００である。可能な限り高い演算精度を保持するとき、指数値の大きい側を左シフトして桁合わせを行う。但し、先行ゼロの桁数より多く左シフトすることは、上位桁が欠落してしまうため、桁合わせに必要な左シフト量Ｅｘ－Ｅｙが先行ゼロの桁数を上回る場合は、指数値の小さい側を右シフトすることで、桁合わせを行う。但し、桁合わせした結果、２つの入力の間に１４桁以上の差がある場合、即ち、重なりが全くない場合には、実際に演算処理を実行しなくとも演算結果を求めることが可能である。ｇｅｔ＿ｚｚは演算結果の下位側を求める演算であり、２入力の重なりがない場合には値の小さい側がそのまま下位側となるからである。

【０１１９】

この条件を満たすか否かを確認するため、（Ｅｘ－Ｌｘ）－（Ｅｙ－Ｌｙ）を計算し、計算結果が１４以上の場合には上記の場合に該当すると判定する。ここで値１４は、使用フォーマットにおける仮数の桁数である。この具体例では上記の場合に該当し、仮数３０１と仮数３０２との間に重なりが存在しない。この場合、仮数３０２がそのままバイパスされて加算結果として取り扱われる。

【０１２０】

更にバイパスされた結果３０３を正規化し、先行ゼロ計数値が１以上の場合はその分だ

け左シフトを行い、桁あふれがある場合には１桁分の右シフトを行う。但し、左シフトした結果指数が奇数になる場合は、左シフト量を１減らす。または右シフトした結果指数が奇数になる場合は、右シフト量を１増やす。指数は、左右シフト量に合わせて調整する。左シフトした場合は左シフト量を指数から減算し、右シフトした場合は右シフト量を指数に加算する。本具体例では１桁左シフトが必要だが、指数が０から－１となり奇数になるので、左シフト量は１減らされて０となり、指数は０のままとなる。このようにして得られた正規化の結果が、そのまま演算結果の仮数３０４及び対応する指数Ｅｚを含む演算結果として出力される。

【０１２１】

図２４は、ｔｗｏ＿ｓｕｍの回路図シンボルを示す図である。上述したｔｗｏ＿ｓｕｍは、ｔｒｉｐｌｅ－ｏｒａｃｌｅｎｕｍ６４計算において頻繁に用いられる演算である。このｔｗｏ＿ｓｕｍ演算、即ちｘとｙとの正確な和を表現するｚとｚｚとを求める演算を、図２４に示すような演算子３１０で表わすものとする。

【０１２２】

図２５は、ｔｒｉｐｌｅ－ｏｒａｃｌｅｎｕｍ６４とｏｒａｃｌｅｎｕｍ６４の和を求める回路の一例を示す図である。図２５に示す回路は、３つのｔｗｏ＿ｓｕｍ演算子３１０と再正規化部３１１とを含む。入力ｂは１つのｏｒａｃｌｅｎｕｍ６４の数であり、入力ａ０，ａ１，ａ２はｔｒｉｐｌｅ－ｏｒａｃｌｅｎｕｍ６４である。また出力ｓ０，ｓ１，ｓ２もまたｔｒｉｐｌｅ－ｏｒａｃｌｅｎｕｍ６４である。図２５に示すような回路構成により、ｔｒｉｐｌｅ－ｏｒａｃｌｅｎｕｍ６４とｏｒａｃｌｅｎｕｍ６４の和を求めることができる。なお再正規化部３１１が実行する演算については、後述する。

【０１２３】

図２６は、ｔｒｉｐｌｅ－ｏｒａｃｌｅｎｕｍ６４同士の和を求めるアルゴリズムを示す図である。前述のｔｗｏ－ｓｕｍを用いて、ｔｒｉｐｌｅ－ｏｒａｃｌｅｎｕｍ６４同士の和を求めることができる。図２６に示すＴｒｉｐｌｅ＿Ａｄｄは、第１のｔｒｉｐｌｅ－ｏｒａｃｌｅｎｕｍ６４であるａ０，ａ１，ａ２と第２のｔｒｉｐｌｅ－ｏｒａｃｌｅｎｕｍ６４であるｂ０，ｂ１，ｂ２との和を求める。この演算は、非特許文献２のＡ．２節に開示されているものと同様である。また非特許文献３に開示されるａｌｇｏｒｉｔｈｍ１３，１４とも同様である。残りの四則演算についても、非特許文献１乃至３に開示の方法を用いて実現可能である。

【０１２４】

以下に、計算結果の再正規化について述べる。再正規化については、非特許文献２のｐ．１１６に述べられており、非特許文献３でもそれを参照している。ここでは、非特許文献２に記載される再正規化をＰｒｉｅｓｔの再正規化と呼ぶ。再正規化前の演算結果（Ｘ０，Ｘ１，Ｘ２，Ｘ３）は、概ね絶対値の大きい順に並んでいるが、一般に、一部の桁にオーバーラップがある。また、Ｘ０がＸ０＋Ｘ１＋Ｘ２＋Ｘ３の最重要（most significant）な部分を固定精度で表現した結果になっていない。

【０１２５】

計算結果（Ｘ０，Ｘ１，Ｘ２，Ｘ３）をＰｒｉｅｓｔの方法で再正規化した結果を（ａ０，ａ１，ａ２）とすると、ａ０＋ａ１＋ａ２は、Ｘ０＋Ｘ１＋Ｘ２＋Ｘ３とｔｒｉｐｌｅの範囲で等しく、且つ、以下を満たす。

【０１２６】

$$|a0| \geqq |a1| \geqq |a2|$$

$$E(i+1) \leqq Ei - 14$$

ここで、Ｅｉは基数を１０とした場合の要素ａｉの指数である。ｏｒａｃｌｅｎｕｍ６４の精度は１０進で１４桁なので、上記の第２の条件は、要素がオーバーラップしていないことを表わす。

【０１２７】

Ｐｒｉｅｓｔの再正規化にも、前述のｔｗｏ＿ｓｕｍが用いられる。まず、オーバーラップを除去するために、図２７のように接続された３つのｔｗｏ＿ｓｕｍ演算子３１０を

用いる。これにより得られる（ｔ０，ｔ１，ｔ２，ｔ３）は、オーバーラップしておらず、且つ、０でない要素については絶対値の大きいものから順に並んでいる。これらをｔ０から順に蓄積していったものを（ａ０，ａ１，ａ２）とすればよい。この蓄積を行なう演算ａｃｃｕｍｕｌａｔｅにもｔｗｏ＿ｓｕｍが用いられる。

【０１２８】

一般に再正規化前の演算結果（Ｘ０，Ｘ１，Ｘ２，Ｘ３）の符号は揃ってなく、前述のようにｔｗｏ＿ｓｕｍを繰り返した結果である（ａ０，ａ１，ａ２）の符号も揃っていない。そこでＰｒｉｅｓｔの再正規化を行った固定精度浮動小数点数の組に含まれる数の符号が全て同符号な場合を、符号の揃ったＰｒｉｅｓｔの再正規化と呼ぶ。

【０１２９】

本願では、Ｐｒｉｅｓｔの再正規化よりも条件の厳しい「強い正規化」を用いる。強い正規化された組（ｂ０，ｂ１，ｂ２）は、次の条件を満たす。なお強い正規化により、ｂ１＝０且つｂ２≠０となる場合が生じるため、｜ｂ０｜＞＝｜ｂ１｜＞＝｜ｂ２｜の条件は一般に成り立たない。

【０１３０】

E(i+1) = Ei - 14

b0,b1,b2：全て同符号

このような強い正規化により、各要素の指数の差と固定小数点数の精度（桁数）とが一致する。従って、強い正規化された組を元のＯｒａｃｌｅ－ｎｕｍｂｅｒ表現に戻すことは容易である。

【０１３１】

まず図２７で示されるオーバーラップが無く０以外の数は絶対値の大きいものから順に並んでいる数の組（ｔ０，ｔ１，ｔ２，・・・）から、符号の揃ったＰｒｉｅｓｔの再正規化された数の組（ａ０，ａ１，ａ２，・・・）を得るための処理を示す。ここでｆｌ＿ｔｒｕｎｃａｔｅ（Ｘ＋Ｙ）は、固定精度浮動小数点数ＸとＹとの和を有効桁で切り捨てる演算を表し、ｔｗｏ＿ｓｕｍ（Ｘ，Ｙ）は前述の通り有効桁に重なりの無い２数を求める演算とする。但しここではｔｗｏ＿ｓｕｍ内での演算は全て四捨五入で丸め処理されるものとする。

【０１３２】

a0 = fl_truncate(t0+t1)

(z0,zz0) = two_sum(t0,t1)

ｚｚ０＝０の場合、以下の様にｚｚ０≠０となるまでｔ２，ｔ３，・・・を順次ａｃｃｕｍｕｌａｔｅしていく。

【０１３３】

a0 = fl_truncate(a0+t2)

(z0,zz0) = two_sum(a0,t2)

入力された数の組(ｔ０，ｔ１，ｔ２，・・・)を全てａｃｃｕｍｕｌａｔｅされてもｚｚ０＝０な場合はそこで処理を終了する。以下ｚｚ０≠０となった後の処理を示す。

【０１３４】

w0 = fl_truncate(z0 - a0)

ここで、ｔｗｏ＿ｓｕｍの性質より、

t0+t1+...+ti = z0+zz0 = a0+w0+zz0

となる。但しｔ０，ｔ１，・・・，ｔｉはｚｚ０≠０となるまでａｃｃｕｍｕｌａｔｅされた数である。この時、以下の通りａ０と（ｗ０＋ｚｚ０）は同符号であり、且つ、ａ０と（ｗ０＋ｚｚ０）に有効桁の重なりは無いと言える。

【０１３５】

（ｉ）ｗ０＝０の場合：ｚｚ０はｆｌ＿ｔｒｕｎｃａｔｅ（ｔ０+ｔ１+...+ｔｉ）によって切り捨てられた数に等しいため、ａ０と（ｗ０＋ｚｚ０）は同符号であり、且つ、ａ０と（ｗ０＋ｚｚ０）に有効桁の重なりは無い。

【０１３６】

(ｉｉ)ｗ０≠０の場合：ｗ０はａ０と同符号でａ０の有効桁最下位にのみ１が立つ数であり、ｚｚ０の絶対値はｗ０の絶対値よりも小さくｚｚ０とｗ０は異符号なため、ａ０と(ｗ０＋ｚｚ０)は同符号であり、且つ、ａ０と（ｗ０＋ｚｚ０）に有効桁の重なりは無い。

【０１３７】

次に、（ｗ０，ｚｚ０，ｔｉ＋１，...）に対して同様の処理を繰り返すことで、ａ１，ａ２，・・・を順次求めることができる。この際、上記（ｉ），（ｉｉ）によりａ１，ａ２，・・・は全て同符合となる。

【０１３８】

図２８は、符号の揃ったＰｒｉｅｓｔの再正規化の例を示す図である。この例においては、簡単のために固定精度浮動小数点数は１０進４桁とし、ｔｗｏ＿ｓｕｍでの丸め処理方法は四捨五入であるとする。図２８において、（ｔ０，ｔ１，ｔ２，ｔ３）はオーバーラップしておらず、且つ、０でない要素については絶対値の大きいものから順に並んでいる。これらをｔ０から順に蓄積していくことで、符号の揃ったＰｒｉｅｓｔの再正規化された数の組ａ０，ａ１，ａ２，ａ３が得られている。

【０１３９】

図２９は、符号の揃ったＰｒｉｅｓｔの再正規化された数の組から強い正規化された数の組を得る計算の一例を示す図である。強い正規化のためには、符号の揃ったＰｒｉｅｓｔの再正規化された数の組から第２要素以降を適切に量子化してやればよい。図２９に示す例では、簡単のために固定精度浮動小数点数は１０進４桁としてある。図２９に示されるように、符号の揃ったＰｒｉｅｓｔの再正規化された数の組（ａ０，ａ１，ａ２）から強い正規化が行われた数の組（ｂ０，ｂ１，ｂ２）が求められている。

【０１４０】

図３０は、量子化のためのｓｃａｌｅ＿ｎｅｘｔ（Ｘ，Ｙ）演算を実現する回路構成の一例を示す図である。図３０に示す回路は、指数補正値レジスタ３２０、入力Ｘレジスタ３２１、入力Ｙレジスタ３２２、内部形式変換回路３２３及び３２４、指数加算器３２５、シフト量演算回路３２６、右シフタ３２７、外部形式変換回路３２８、及び出力Ｚレジスタ３２９を含む。この回路の入力データは以下の条件を満たしていることを前提としている。

【０１４１】

指数Ｙ ≧ 指数Ｘ ＋ ｔ

ここでｔはｏｒａｃｌｅｎｕｍ６４の桁数である。本実施例では、ｔは例えば１４である。

【０１４２】

内部形式変換回路３２３及び３２４により、入力を指数部及び仮数部に分割し、入力の値表現を内部形式に変換する。入力Ｘの指数及び仮数をそれぞれ指数Ｘ及び仮数Ｘとする。入力Ｙの指数を指数Ｙとする。指数補正値レジスタ３２０には、予め決められた固定値ｔが格納される。レジスタ格納値として設定するのではなく、ハード的に固定値を設定してもよい。

【０１４３】

シフト量演算回路３２６が指数Ｘと指数Ｙと固定値ｔを受け取る。シフト量演算回路３２６は、指数Ｙ－指数Ｘ－ｔの演算結果をシフト量として出力する。右シフタ３２７は、このシフト量と仮数Ｘとを受け取り、仮数Ｘを当該シフト量だけ右シフトした結果を出力する。シフトアウトした桁は捨てられる。

【０１４４】

指数加算器３２５は、指数Ｙと固定値tとを受け取り、指数Ｙ－ｔの演算結果を出力する。外部形式変換回路３２８は、指数加算器３２５からの指数と右シフタ３２７からの仮数を外部形式に変換し、出力Ｚレジスタ３２９に出力する。

【０１４５】

図３１は、仮数部を内部形式に変換せずにｓｃａｌｅ＿ｎｅｘｔ演算を行う回路の構成の一例を示す図である。図３１において、図３０と同一の構成要素は同一の番号で参照し

、その説明は適宜省略する。図３１では、内部形式変換回路３２３の代りに内部形式変換回路３２３Ａが設けられ、外部形式変換回路３２８の代りに外部形式変換回路３２８Ａが設けられている。入力データがｏｒａｃｌｅｎｕｍ６４形式の場合、右シフタはシフト量２につき８ビットシフトする。

【０１４６】

内部形式変換回路３２３Ａ及び３２４により、入力を指数部及び仮数部に分割し、入力の値表現を内部形式に変換する。入力Ｘの指数を指数Ｘとする。入力Ｙの指数を指数Ｙとする。指数補正値レジスタ３２０には、予め決められた固定値ｔが格納される。レジスタ格納値として設定するのではなく、ハード的に固定値を設定してもよい。

【０１４７】

シフト量演算回路３２６が指数Ｘと指数Ｙと固定値ｔを受け取る。シフト量演算回路３２６は、指数Ｙ－指数Ｘ－ｔの演算結果をシフト量として出力する。右シフタ３２７は、このシフト量と仮数Ｘとを受け取り、仮数Ｘを当該シフト量だけ右シフトした結果を出力する。シフトアウトした桁は捨てられる。

【０１４８】

指数加算器３２５は、指数Ｙと固定値tとを受け取り、指数Ｙ－ｔの演算結果を出力する。外部形式変換回路３２８Ａは、指数加算器３２５からの指数を外部形式に変換し、出力Ｚレジスタ３２９に出力する。また右シフタ３２７の出力する右シフト後の仮数は、そのまま出力Ｚレジスタ３２９に出力される。

【０１４９】

以下に、ｔｒｉｐｌｅ－ｏｒａｃｌｅｎｕｍ６４形式に対する丸め処理を考える。ここでは、丸め処理対象であるｔｒｉｐｌｅ－ｏｒａｃｌｅｎｕｍ６４形式の数の組（ａ０，ａ１，ａ２）は再正規化されていることを前提とする。なおこの前提条件は、Ｐｒｉｅｓｔの再正規化及び強い正規化の何れであってもよい。

【０１５０】

図３２は、Ｏｒａｃｌｅ－Ｄａｔａｂａｓｅ（商標）におけるＮＵＭＢＥＲ型の３種類の精度指定方法を示す表である。ここで、丸め方向は常に四捨五入である。例えば、計算結果が １２３４．５６だった場合に、ＮＵＭＢＥＲ（４）、ＮＵＭＢＥＲ（４，－２）、ＮＵＭＢＥＲ（４，１）のそれぞれの指定に対する丸め結果は、以下の通りとなる。

【０１５１】

ＮＵＭＢＥＲ（４）→１２３４
ＮＵＭＢＥＲ（４，－２）→１２００
ＮＵＭＢＥＲ（４，１）→エラー

最後の例でエラーとなるのは、計算結果である１２３４．５６を小数点１位まで表わすように丸めた結果である１２３４．６が、４桁の精度内に収まらないためである。

【０１５２】

ｔｒｉｐｌｅ－ｏｒａｃｌｅｎｕｍ６４形式においても、このような丸め処理を、エラー判定も含めて効率良く実現する必要がある。ここで用いることのできる演算器は、８バイト幅のオペランドを２つの入力として、１つの８バイト幅の数を出力する演算器である。丸め処理においても、このような演算器を前提とする。

【０１５３】

そこで、ｔｒｉｐｌｅ－ｏｒａｃｌｅｎｕｍ６４で表現された数の丸め処理を、以下の３ステップで実現することにする。

【０１５４】

１）四捨五入が発生する桁位置のみ５があるような数を生成する。

【０１５５】

２）ｔｒｉｐｌｅ－ｏｒａｃｌｅｎｕｍ６４形式の数に、上で生成した数を加える。

【０１５６】

３）適切な桁位置で切捨てを行う。

まず、第１ステップのアルゴリズムについて述べる。以降、丸め対象数のｔｒｉｐｌｅ－

ｏｒａｃｌｅｎｕｍ６４表現を（ａ０，ａ１，ａ２）とする。また、この組の最重要要素ａ０の値は、

a0 = M * 100^e

で表わされるとする。また、上式のｅを求めるための演算をｅ（ａ０）と表すことにする。

【０１５７】

第１ステップは、精度指定方法により、生成アルゴリズムが異なる。精度指定がない場合（引数なしのＮＵＭＢＥＲ指定の場合）には、仮数部が２０バイト以下となるように丸めが行われる必要がある。

【０１５８】

精度指定時のアルゴリズムを記述するには、ａ０を１０進表記で表した方が都合がよい。そこで、

a0 = M' * 10^e'

の形で表す。また、仮数部が１０＞｜Ｍ'｜＞＝０を満たすように正規化されているものとする。また、前と同様に、ｅ'を求めるための演算をｅ'（ａ０）と表す。

【０１５９】

図３３は、四捨五入が発生する桁位置のみ５があるような数を生成するアルゴリズムの例を示す図である。図３３（ａ）及び（ｂ）には、１０進桁数による精度指定の場合のアルゴリズムが示される。これら２つのアルゴリズムは、いずれも２入力１出力型の演算器として実現可能である。図３３（ｃ）には、スケール（小数点からの相対位置）指定の場合のアルゴリズムが示される。小数点からの相対位置を指定するので、ａ０に依存しない計算となる。この図３３（ｃ）に示すアルゴリズムは、図示した通りの数式をそのままプログラム化してもよいが、ｎの各値に対応するテーブルを用意しておいて、ｎをキーにテーブルを引くようなプログラムを用意してもよい。

【０１６０】

スケール指定があった場合に、丸めた結果が指定精度に収まるか否かは、以下のように判定できる。ＮＵＭＢＥＲ（ｐ，ｓ）が指定された場合、まず、上述の精度指定の場合のアルゴリズムとスケール指定のアルゴリズムとを両方用いて、ｐ＿ｄ及びｐ＿ｓの２つの値を計算する。

【０１６１】

p_d = get_comma5(a0, digits=p)
p_s = get_comma5(scale=s)

このとき、｜ｐ＿ｄ｜＞｜ｐ＿ｓ｜となった場合、すなわちｐ＿ｄの絶対値がｐ＿ｓの絶対値よりも大きい場合には、スケール指定で丸められた結果を表現するためには、精度が不足していることがわかる。

【０１６２】

次に第２ステップでは、第１ステップで生成した数を、ｔｒｉｐｌｅ－ｏｒａｃｌｅｎｕｍ６４で表現された数に加える。この加算には、図２５に示したアルゴリズムが利用できる。

【０１６３】

第３ステップでは、適切な位置で切り捨てを行う。ここでも、第１ステップで生成した数ｐ（精度指定の方式に応じたｐ＿ｄまたはｐ＿ｓの何れかの値）を利用することができる。この数ｐは、四捨五入のために加算される数であり、丸めが発生する桁位置に関する情報を完全に含んでいる。そのため、ｔｒｉｐｌｅ－ｏｒａｃｌｅｎｕｍ６４の各要素とｐとの２つのオペランドを入力として、各要素を適切にｔｒｕｎｃａｔｅする演算を定義することができる。

【０１６４】

第２ステップの計算結果を（ｂ０，ｂ１，ｂ２）とすると、丸めの第３段階の結果（ｃ０，ｃ１，ｃ２）は、

c0 = truncate(b0, p)

```
  c1 = truncate(b1, p)
  c2 = truncate(b2, p)
```
となる。
【０１６５】
図３４は、ｇｅｔ＿ｃｏｍｍａ５（ｐｒｅｃｉｓｉｏｎ）を実現する回路構成の一例を示す図である。図３４に示す回路は、精度ｐレジスタ３３０、入力Ｘレジスタ３３１、内部形式変換回路３３２、先行ゼロ計数回路３３３、指数加算器３３４、レジスタ３３５及び３３６、セレクタ３３７、外部形式変換回路３３８、及び出力Ｚレジスタ３３９を含む。
【０１６６】
内部形式変換回路３３２により、入力を指数部及び仮数部に分割し、入力の値表現を内部形式に変換する。入力Ｘの指数と仮数をそれぞれ指数Ｘ、仮数Ｘとする。先行ゼロ計数回路３３３は、仮数Ｘを受け取り、仮数Ｘ中の先行ゼロの数を計数する。指数加算器３３４は、指数Ｘ＋１－精度ｐ－仮数Ｘの先行ゼロ計数値を計算する。セレクタ３３７は、指数加算器３３４の出力が奇数の場合にはレジスタ３３６の５０００－００を選択し、指数加算器３３４の出力が偶数の場合にはレジスタ３３５の０５００－００を選択する。
【０１６７】
指数加算器３３４から外部形式変換回路３３８へ供給される指数は、指数加算器３３４の出力が奇数の場合は当該指数の最下位ビットをゼロとし、指数加算器３３４の出力が偶数の場合はそのままとする。外部形式変換回路３３８は、内部形式変換回路３３２からの符号、指数加算器３３４からの指数、及びセレクタ３３７からの仮数を外部形式の数に変換して、変換結果を出力Ｚレジスタ３３９に出力する。
【０１６８】
図３５は、ｔｒｕｎｃａｔｅ演算を実現する回路構成の一例を示す図である。図３５の回路は、入力Ｘレジスタ３４０、ｃｏｍｍａ５レジスタ３４１、内部形式変換回路３４２及び３４３、先行ゼロ計数回路３４４、マスク値生成回路３４５、マスク回路３４６、外部形式変換回路３４７、及び出力Ｚレジスタ３４８を含む。
【０１６９】
内部形式変換回路３４２及び３４３により、入力を符号、指数部、及び仮数部に分割し、入力の値表現を内部形式に変換する。入力Ｘの符号、指数、及び仮数をそれぞれ符号Ｘ、指数Ｘ、及び仮数Ｘとする。入力ｃｏｍｍａ５の指数と仮数をそれぞれ指数c及び仮数cとする。
【０１７０】
先行ゼロ計数回路３４４は、仮数ｃを受け取り、仮数ｃ中の先行ゼロの数を計数する。マスク値生成回路３４５は、指数Ｘ、指数c、及び仮数cの先行ゼロ計数値を入力として受け取り、これらの入力に応じてマスクデータを生成する。マスク回路３４６は、仮数Ｘをマスクデータに応じてマスクし、マスク後の値を仮数として出力する。外部形式変換回路３４７は、内部形式変換回路３４２からの符号Ｘ及び指数Ｘ並びにマスク回路３４６からの仮数を外部形式の数に変換して、変換結果を出力Ｚレジスタ３４８に出力する。
【０１７１】
図３６は、マスク値生成回路の構成の一例を示す図である。図３６に示すマスク値生成回路３４５は、マスク桁演算回路３５０、デコーダ３５１、及びセレクタ３５２－１乃至３５２－１４を含む。マスク桁演算回路３５０は、指数Ｘ、指数ｃ、及び先行ゼロ計数値Ｌｃを受け取り、マスク桁を演算する。マスク桁は、使用フォーマットの仮数の桁数を１４とすると、１４＋（指数ｃ－Ｌｃ）－指数Ｘで求められる。この値は、マスクデータの最下位桁から何桁目までがゼロに設定されるかを示す。このように計算されたマスク桁演算結果に基づいて、デコーダ３５１が、桁ごとのセレクト信号を生成する。デコーダ３５１が生成したセレクト信号は、１４桁にそれぞれ対応するセレクタ３５２－１乃至３５２－１４に供給される。マスク桁演算結果がｎの場合、下位からｎ桁は０００００側を選択するようにセレクト信号が生成される。マスク桁演算結果が０の場合は、全ての桁が１１１

１側を選択するようにセレクト信号が生成される。このセレクト信号に応じて、セレクタ３５２－１乃至３５２－１４が、１１１１又は０００００の何れかのビット列を選択し、選択したビット列をマスクデータとして出力する。

【０１７２】

図３７は、ｐ＿ｄ及びｐ＿ｓの絶対値大小比較エラー判定回路の構成の一例を示す図である。この回路は、ｐ＿ｓレジスタ３６０、ｐ＿ｄレジスタ３６１、内部形式変換回路３６２及び３６３、指数仮数演算回路３６４、セレクタ３６５及び３６６、シフタ３６７及び３６８、符号演算器３６９、加算器３７０、コンプリメンタ３７２、コンプリメンタ３７３、及びエラーフラグレジスタ３７１を含む。

【０１７３】

内部形式変換回路３６２及び３６３により、入力を指数部、及び仮数部に分割し、入力の値表現を内部形式に変換する。入力ｐ＿ｓの指数、仮数をそれぞれ指数Ｘ、仮数Ｘとする。入力ｐ＿ｄの指数、仮数をそれぞれ指数Ｙ、仮数Ｙとする。

【０１７４】

指数仮数演算回路３６４が、指数Ｘ及び指数Ｙと、仮数Ｘ及び仮数Ｙとを受け取る。指数仮数演算回路３６４は、指数Ｘと指数Ｙの大小比較をする。大小比較の結果に基づいて、指数仮数演算回路３６４は、指数の大きい側の仮数（第１の仮数）がシフタ３６７に入力され、且つ、指数の小さい側の仮数（第２の仮数）がシフタ３６８に入力されるように、セレクト信号を生成する。指数仮数演算回路３６４は、指数Ｘと指数Ｙとの差の絶対値と第１の仮数の先行ゼロ計数値とを比較し、前者の方が大きい場合は、後者をシフタ３６７の左シフト量として出力する。前者の方が小さい場合は、指数仮数演算回路３６４は、前者をシフタ３６７の左シフト量として出力する。

【０１７５】

指数仮数演算回路３６４は、指数Ｘと指数Ｙとの差の絶対値と第１の仮数の先行ゼロ計数値とを比較し、前者の方が大きい場合は、前者から後者を減算した値をシフタ３６８の右シフト量として出力する。前者の方が小さい場合は、指数仮数演算回路３６４は、ゼロをシフタ３６８の右シフト量として出力する。

【０１７６】

シフタ３６７は、入力されたシフト量に基づき、入力された仮数を左シフトする。シフタ３６８は、入力されたシフト量に基づき、入力された仮数を右シフトする。各シフタのシフト結果はコンプリメンタ３７２、コンプリメンタ３７３に入力される。

【０１７７】

コンプリメンタ３７２は入力された前述のセレクト信号が反転された信号により入力された左シフトの出力をコンプリメントするＥＯＲ回路（排他的論理和回路）である。コンプリメンタ３７３は入力された前述のセレクト信号により入力された右シフトの出力をコンプリメントするＥＯＲ回路（排他的論理和回路）である。各コンプリメンタの結果は加算器３７０に入力される。

【０１７８】

加算器３７０にキャリーが入力される。加算器３７０による加算の結果桁あふれが生じた場合は、符号演算器３６９にキャリーアウトが供給される。

【０１７９】

符号演算器３６９は、加算器３７０からキャリーアウトを受け取る。符号演算器３６９は、キャリーアウトの反転値が１であるとき、エラーフラグを１にする。それ以外の場合において、エラーフラグは０である。符号演算器３６９が生成するエラーフラグは、エラーフラグレジスタ３７１に格納される。

【０１８０】

図３８は、上記の符号演算器の回路構成の一例を示す図である。図３８（ａ）に示す符号演算器は、インバータ３８０を含む。この回路により、キャリーアウトの反転値が１のとき、エラーフラグが１になる。図３８（ｂ）に示す表は、ｐ＿ｄとｐ＿ｓの絶対値大小比較エラー判定回路に入力されたｐ＿ｄとｐ＿ｓの絶対値の大小関係と、その場合に出力

される各信号の対応表である。ここで｜ｐ＿ｄ｜はｐ＿ｄの絶対値である。ｐ＿ｓについても同様である。

【０１８１】

図３９は、Ｏｒａｃｌｅ－ｎｕｍｂｅｒの長さを求める演算の一例を示す図である。前述の演算の結果として得られたｔｒｉｐｌｅ－ｏｒａｃｅｌｎｕｍ６４形式の数を、１つのＯｒａｃｌｅ－ｎｕｍｂｅｒとしてメモリに書き込む際には、このＯｒａｃｌｅ－ｎｕｍｂｅｒの長さを求める必要がある。Ｏｒａｃｌｅ－ｎｕｍｂｅｒの長さを求めるには、図３９に示されるようにすればよい。但し、ｌｅｎｇｔｈ（ａＸ）は、ｏｒａｃｌｅｎｕｍ６４形式での有効桁数を求める演算とする。ａ２が０であればＯｒａｃｌｅ－ｎｕｍｂｅｒの長さは短くなるし、更にａ１が０であればＯｒａｃｌｅ－ｎｕｍｂｅｒの長さは更に短くなる。

【０１８２】

強い再正規化を施したｔｒｉｐｌｅ－ｏｒａｃｅｌｎｕｍ６４形式において、ａ０、ａ１、ａ２は、１つのＯｒａｃｌｅ－ｎｕｍｂｅｒの仮数部を７バイト、７バイト、６バイトの３つの部分に分割した各部を表現している。従って、基本的には、ａ０、ａ１、ａ２の仮数部を連結するだけで、Ｏｒａｃｌｅ－ｎｕｍｂｅｒ表現が得られる。但し、ａ０、ａ１の長さが短い場合には、後続ゼロを拡張してからメモリに格納する必要がある。

【０１８３】

図４０は、ｅｘｐａｎｄ演算を実現する回路の構成の一例を示す図である。図４０の回路は、入力Ｘレジスタ３９０、内部形式変換回路３９１、後続ゼロ計数回路３９２、桁セレクト演算回路３９３、及び出力Ｚレジスタ３９４を含む。

【０１８４】

内部形式変換回路３９１は、入力Ｘレジスタ３９０の格納値の仮数部を受け取り、内部形式の仮数部を出力する。後続ゼロ計数回路３９２は、内部形式の仮数部を受け取り、受け取った仮数部から後続ゼロの数を求め、後続ゼロの計数結果を基にした桁セレクトデータを出力する。桁セレクト演算回路３９３は、入力Ｘレジスタ３９０から格納値の符号部と仮数部とを受け取り、更に後続ゼロ計数回路３９２から桁セレクトデータを受け取る。符号と桁セレクトデータとに基づいて、桁セレクト演算回路３９３は、受け取った仮数又はｏｒａｃｌｅｎｕｍ６４のゼロ（０ｘ０１または０ｘ６５）の何れかを桁毎に選択して出力する。後続ゼロ計数回路３９２において後続ゼロの部分であると判断された桁においては、ゼロが選択されて出力される。出力されたデータは出力Ｚレジスタ３９４に格納される。

【０１８５】

図４１は、上記の後続ゼロ計数回路の構成の一例を示す図である。図４１（ａ）に示すように、後続ゼロ計数回路は、変換回路４００を含む。変換回路４００は仮数を入力データとして受け取り、図４１（ｂ）に示すテーブルに従い入力データから出力データを生成する。この出力データが後続ゼロ計数値であり、２進数により計数値が表わされる。テーブルでは、一番右側のＸはゼロ以外の値を表し、それ以外のＸはドントケアである。０は計数対象となるゼロを表わす。

【０１８６】

図４２は、桁セレクト演算回路の構成の一例を示す図である。図４２に示す桁セレクト演算回路３９３は、デコーダ４１０及びセレクタ４１１－１乃至４１１－１４を含む。デコーダ４１０は、符号と後続ゼロ計数値とを受け取り、桁毎の選択信号である桁セレクト信号を生成する。デコーダ４１０が生成した桁セレクト信号は、例えば１４桁にそれぞれ対応するセレクタ４１１－１乃至４１１－１４に供給される。後続ゼロ計数値がｎの場合、下位からｎ桁は０ｘ０１又は０ｘ６５側を選択するように桁セレクト信号が生成される。このとき、符号が正を表す１のときは０ｘ０１が選択され、符号が負を表す０のときは０ｘ６５が選択される。下位より数えてｎ＋１桁から上位の桁は、入力の仮数がそのまま選択される。このようにして桁毎に選択された入力仮数、０ｘ０１、又は０ｘ６５が、仮数データとして出力される。

【０１８７】
図４３は、固定精度浮動小数点加減算器の構成の一例を示す図である。図４３に示す固定精度浮動小数点加減算器は、図５に示す固定精度浮動小数点加減算器に対して、ここまでに説明した種々の演算の機能を纏めて付加することにより得られる回路であり、図２の演算回路１１９の一部分に相当する。ここで付加される種々の演算は、図１５のｇｅｔ＿ｚ演算、図２０のｇｅｔ＿ｚｚ演算、図３０のｓｃａｌｅ＿ｎｅｘｔ演算、図３４のｇｅｔ＿ｃｏｍｍａ５演算、図３５のｔｒｕｎｃａｔｅ演算、図３７のエラー判定、及び図４０のｅｘｐａｎｄ演算を含む。図４３に示す固定精度浮動小数点加減算器は、値レジスタ４５０、入力Ｘレジスタ４５１、入力Ｙレジスタ４５２、内部形式変換回路４５３及び４５４、後続ゼロ計数回路４５５、指数仮数マスク演算回路４５６、セレクタ４５７及び４５８を含む。図４３に示す固定精度浮動小数点加減算器は、更に、シフタ４５９及び４６０、桁セレクト演算回路４６１、マスク回路４６２、符号指数演算回路４６３、絶対値加算器４６４、正規化回路４６５、セレクタ４６６乃至４６８、正規化回路４６９を含む。図４３に示す固定精度浮動小数点加減算器は、更に、セレクタ４７０及び４７１、丸め回路４７２、外部形式変換回路４７３、セレクタ４７４及び４７５、エラーフラグレジスタ４７６、及び出力Ｚレジスタ４７７を含む。

【０１８８】
図４３に示す固定精度浮動小数点加減算器の各部は、前述の演算回路の対応する各部に相当する。例えば、後続ゼロ計数回路４５５及び桁セレクト演算回路４６１は、図４０のｅｘｐａｎｄ演算の後続ゼロ計数回路３９２及び桁セレクト演算回路３９３に相当する。また例えばマスク回路４６２は、図３５のｔｒｕｎｄａｔｅ演算のマスク回路３４６に相当する。また例えば正規化回路４６５は、図２０のｇｅｔ＿ｚｚ演算の正規化回路２７０に相当する。例えば正規化回路４６９は、図１５のｇｅｔ＿ｚ演算及び図２０のｇｅｔ＿ｚｚ演算の正規化回路２３４に相当する。また例えばセレクタ４６８は、図３４のｇｅｔ＿ｃｏｍｍａ５演算のセレクタ３３７に相当する。また指数仮数マスク演算回路４５６及び符号指数演算回路４６３は、各演算回路の対応する回路部を纏めたものに相当する。これら回路部分の動作については、前述の各演算回路の対応する回路部分の動作と同様である。なお丸め回路４７２を通す必要のない演算もあるが、セレクタの数を削減するために、この構成例では全ての演算結果を丸め回路４７２に通す形態としている。丸め処理の必要がない演算については、丸めモードを強制的にゼロ方向に設定することで、丸め回路４７２を通さなかった場合と同等の結果を得ることができる。

【０１８９】
図４４は、指数仮数マスク演算回路の構成の一例を示す図である。図４４に示す指数仮数マスク演算回路４５６は、比較回路４８０、絶対値加算器４８１、セレクタ４８２乃至４８６、先行ゼロ計数回路４８７及び４８８、加算器４９１及び４９２、及びマスク生成回路４９３を含む。図４４に示す指数仮数マスク演算回路４５６は更に、セレクタ４９４乃至４９６、及び加算器４９７を含む。

【０１９０】
比較回路４８０、絶対値加算器４８１、先行ゼロ計数回路４８７、セレクタ４９４及び４９６、及び加算器４９２は、図６に示す比較回路１５１、絶対値加算器１５２、先行ゼロ計数回路１５６、セレクタ１５８及び１５７、及び加算器１５５に相当する。但し、絶対値加算器４８１は、指数Ｘ及び指数Ｙに加え、各演算機能に応じた値を受け取り、所定の加減算を実行する。絶対値加算器４８１は、ｇｅｔ＿ｚ演算やｇｅｔ＿ｚｚ演算の場合には、指数Ｘと指数Ｙの差の絶対値を計算する。また例えばｓｃａｌｅ＿ｎｅｘｔ演算の場合であれば、絶対値加算器４８１は、指数Ｙ－指数Ｘ－ｔ（ｏｒａｃｌｅｎｕｍ６４の桁数）の演算結果をシフト量として出力する。また加算器４９１は、図３６のマスク桁演算回路３５０の機能を実現し、指数Ｘ、指数ｃ（ｃｏｍｍａ５の指数）、及び先行ゼロ計数値Ｌｃを受け取り、マスク桁を演算する。マスク生成回路４９３は、図３６のデコーダ３５１及びセレクタ３５２－１乃至３５２－１４に相当し、上記マスク桁に応じてマスクデータを生成する。また加算器４９１は更に、図１５及び図２０の指数仮数演算回路の機

能を実現し、（指数Ｘ－仮数Ｘの先行ゼロ計数値）－（指数Ｙ－仮数Ｙの先行ゼロ計数値）の絶対値が１４以上であるか否かを判定する。この絶対値が１４以上である場合、加算器４９１は、バイパス経路を選択するためのバイパスセレクト信号を生成する。

【０１９１】

以上、本発明を実施例に基づいて説明したが、本発明は上記実施例に限定されるものではなく、特許請求の範囲に記載の範囲内で様々な変形が可能である。

【符号の説明】

【０１９２】

１１０　プロセッサ
１１１　メモリ
１１２　２次キャッシュ部
１１３　１次キャッシュ部
１１４　制御部
１１５　演算部
１１６　レジスタ
１１７　演算制御部
１１８　演算器
１１９　演算回路

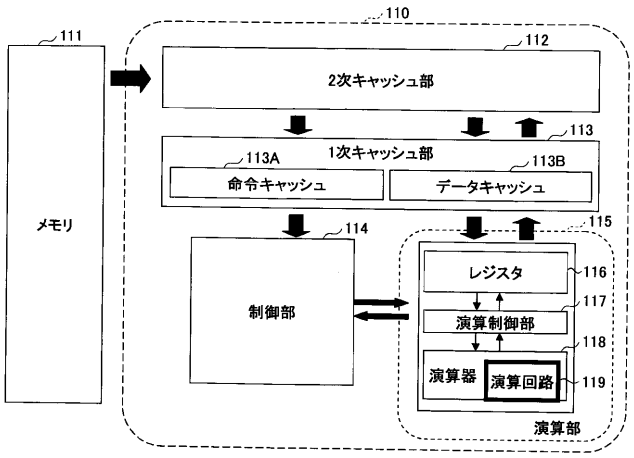
【図１】

Oracle－numberの具体例の表を示す図

		exp	byte[0]	length	Oracle NUMBER representation byte[0]	byte[1]	byte[2]	...	byte[20]	Number
Positive	Inf	—	255	2	255	101				Inf
	MAX Number	+124	255	21	255	100	100	...	100	99.999999999999999999999999999999999999E+124
			255	2	255	11	100	...		10E+124
			255	2	255	2	100	...		10E+123
	...	...	...	...	...	...	...	...		...
			194	2	194	11	...	...		10E+2
			194	2	194	2	...	...		10E+1
		+0	193	2	193	11	...	...		10E+0
		...	...	...	...	...	...	...		...
Zero		130	128	2	128	2	...	...		1E-130
		—	128	1	128					0
Negative		130	127	3	127	91	102			-10E-130
			126	3	126	100	102			-10E-129
			126	3	126	91	102			-10E-128
		...	...	...	...	...	...	...	...	...
			3	3	1	91	102			-10E+122
			3	3	0	100	102			-10E+123
			3	21	0	91	102		2	-10E+124
	MIN Number	+124	0	21	0	2	2	...	...	99.999999999999999999999999999999999999e+124
	-Inf	—	0	1	0					-Inf

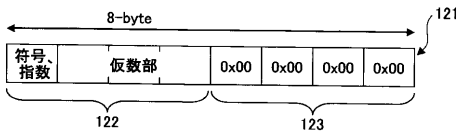
【図２】

コンピュータシステムの構成の一例を示す図



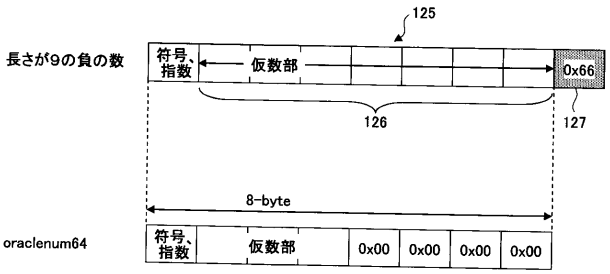
【図３】

oraclenum64の構成を示す図



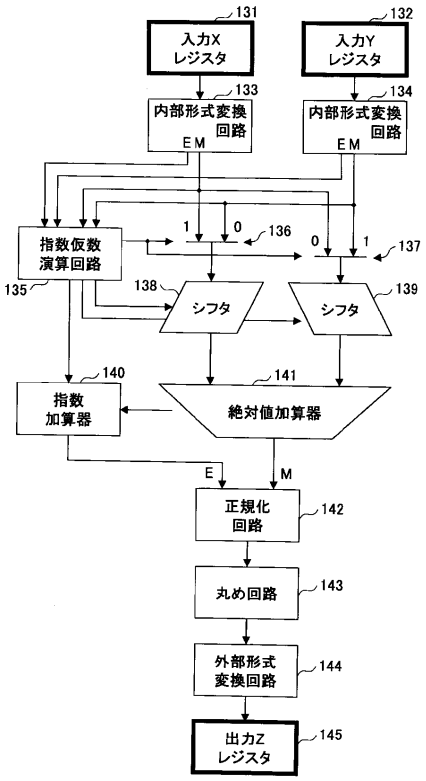
【図４】

長さが9バイトのoraclenum64の構成を示す図



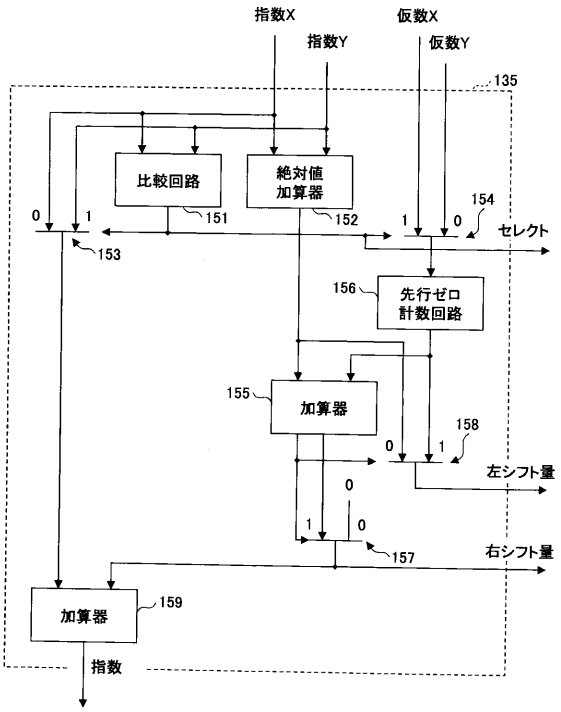
【図５】

oraclenum64を直接演算対象とできる演算器の構成の一例



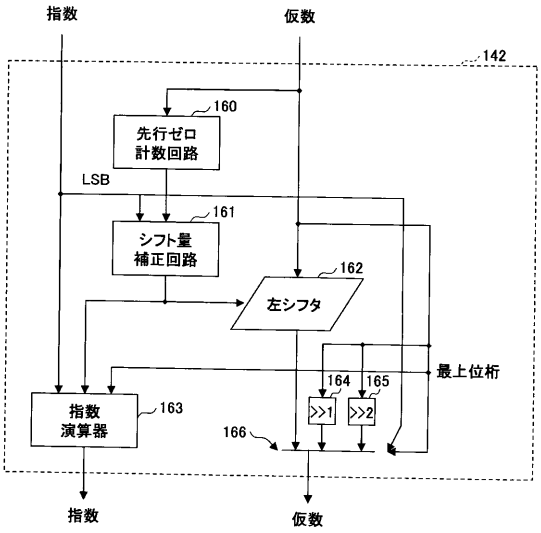
【図６】

指数仮数演算回路の構成の一例を示す図



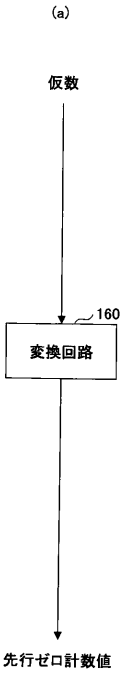
【図７】

正規化回路の構成の一例を示す図



【図８】

先行ゼロ計数回路の構成の一例を示す図

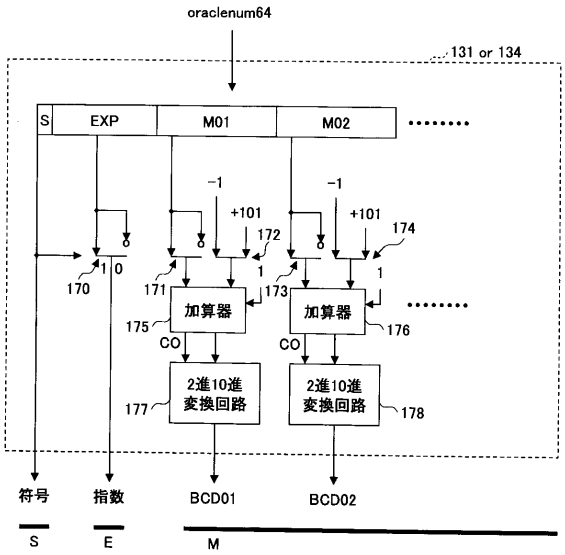


(b)

入力	出力
××××××××××××××	0000
0×××××××××××××	0001
00××××××××××××	0010
000×××××××××××	0011
0000××××××××××	0100
00000×××××××××	0101
000000××××××××	0110
0000000××××××××	0111
00000000×××××××	1000
000000000××××××	1001
0000000000××××××	1010
00000000000××××	1011
000000000000××	1100
0000000000000×	1101
0000000000000	1110

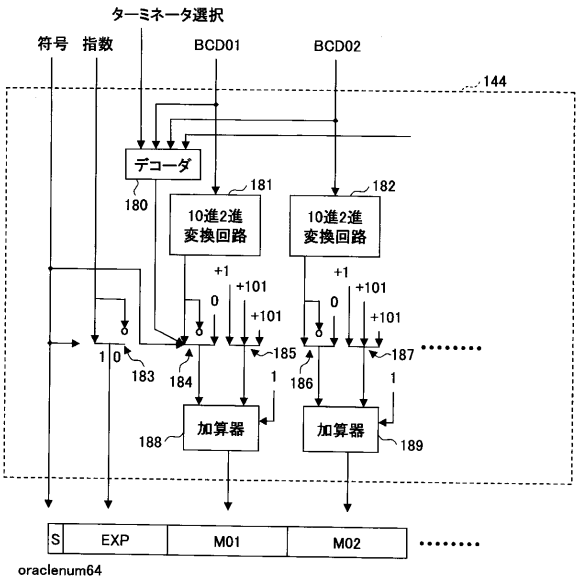
【図９】

内部形式変換回路の構成の一例を示す図



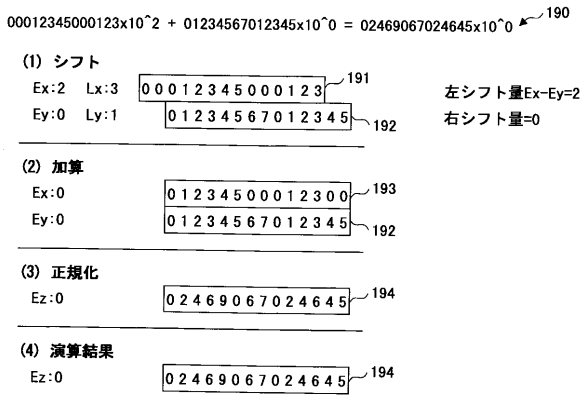
【図１０】

外部形式変換回路の構成の一例を示す図



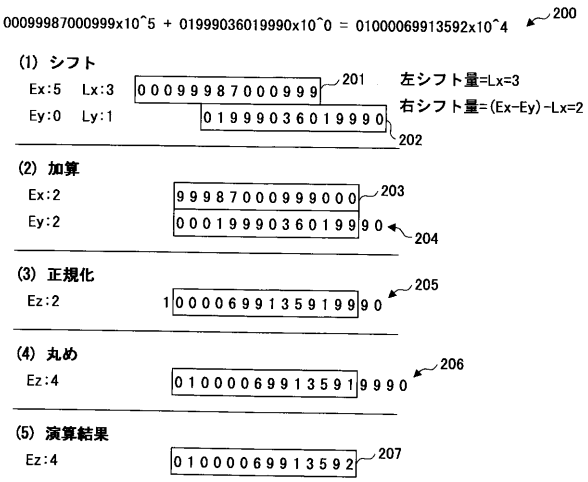
【図１１】

固定精度浮動小数点数加算の具体例を示す図



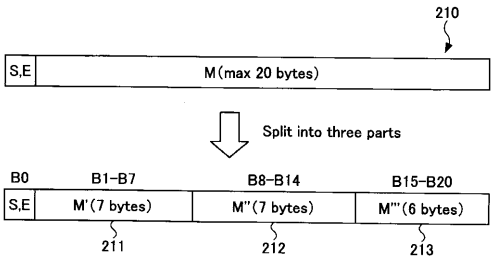
【図１２】

固定精度浮動小数点数加算の別の具体例を示す図



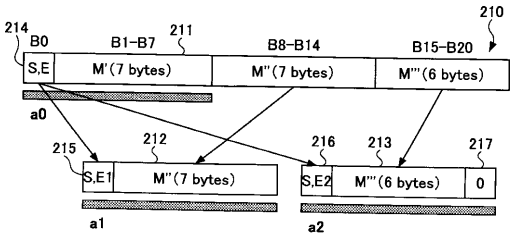
【図１３】

21バイト長のOracle－numberを3つの部分に分ける方法を示す図



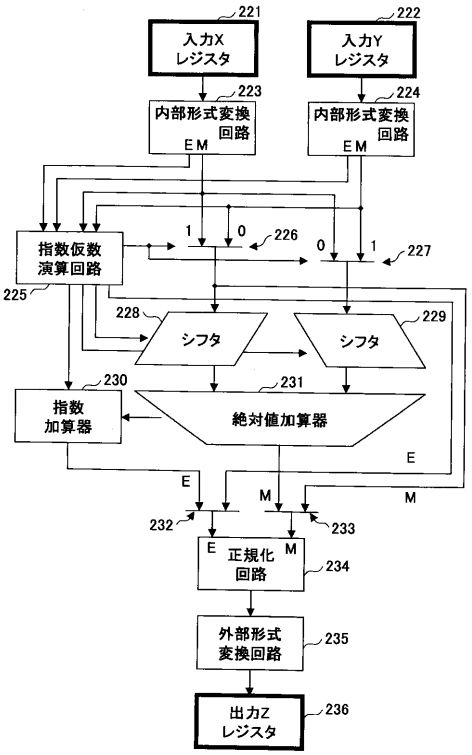
【図１４】

分割された3つの仮数部に
それぞれ対応するoraclenum64数を生成する方法を示す図



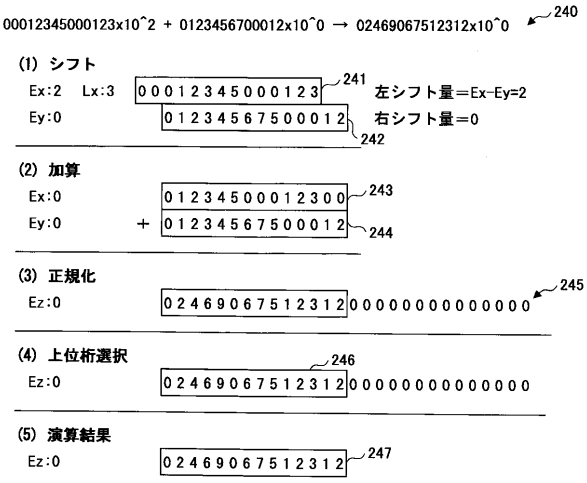
【図１５】

get__z演算を実現する回路の構成の一例を示す図



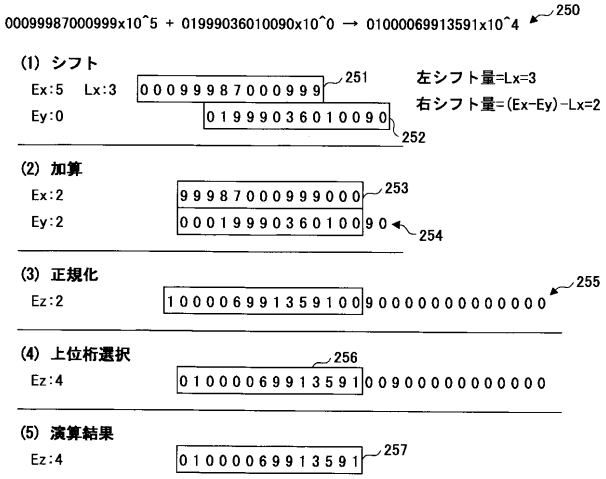
【図１６】

get__z演算の具体例を示す図



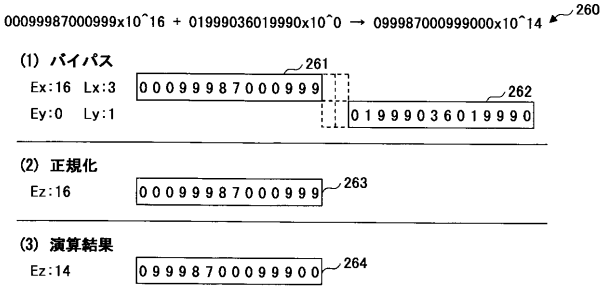
【図１７】

get__z演算の別の具体例を示す図



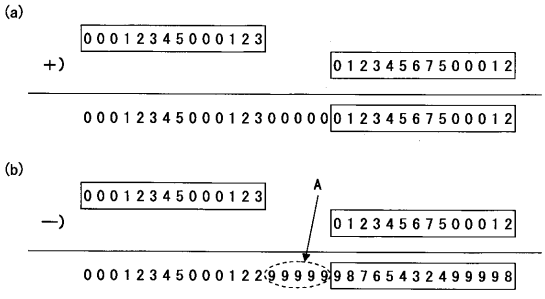
【図１８】

get__z演算の更に別の具体例を示す図



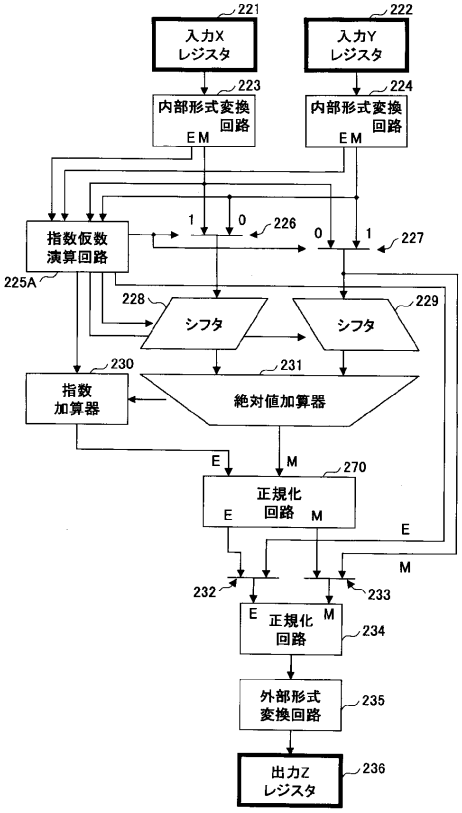
【図１９】

2つの入力値の絶対値がフォーマットの桁数以上離れている場合の加算演算の例を示す図



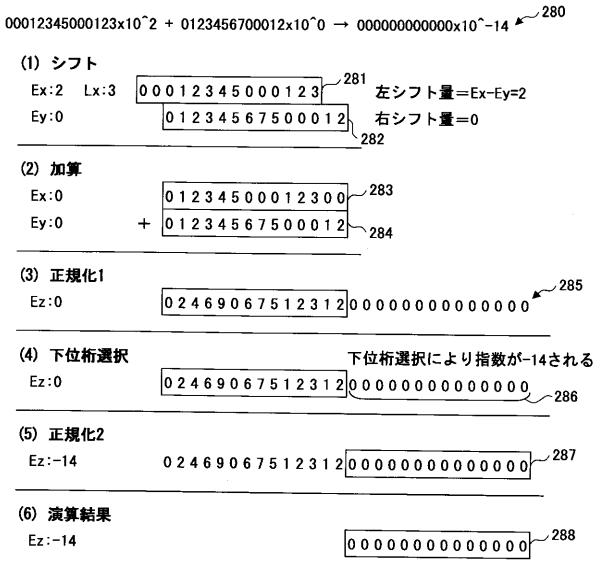
【図２０】

get__zz演算を実現する回路の構成の一例を示す図



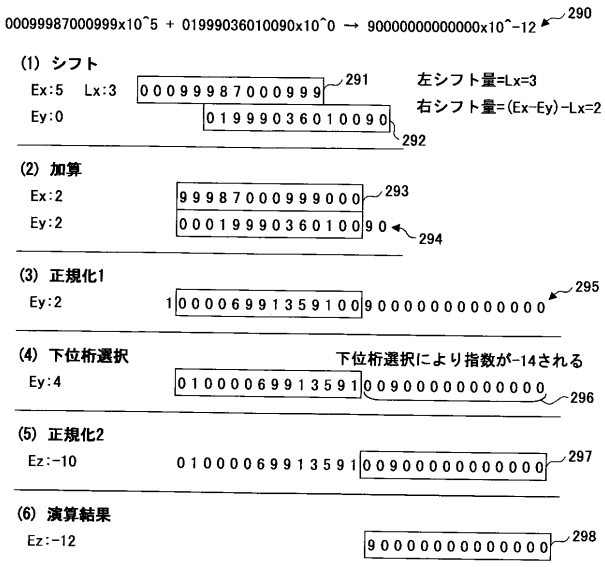
【図２１】

get__zz演算の具体例を示す図



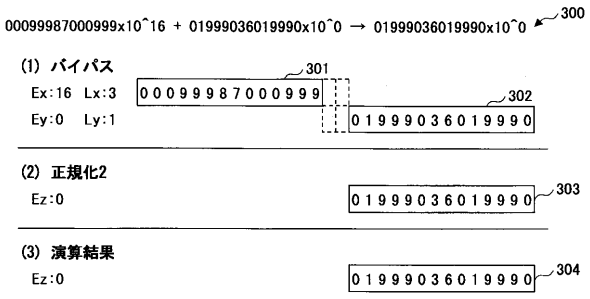
【図２２】

get__zz演算の別の具体例を示す図



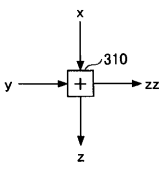
【図２３】

get__zz演算の更に別の具体例を示す図



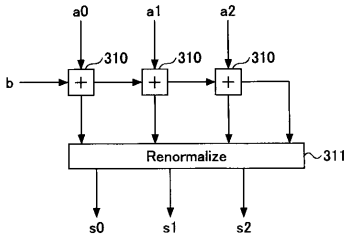
【図２４】

two__sumの回路図シンボルを示す図



【図２５】

triple－oraclenum64とoraclenum64の和を求める回路の一例を示す図



【図２６】

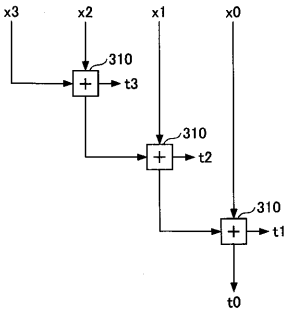
triple－oraclenum64同士の和を求めるアルゴリズムを示す図

```
Accumulate(u, v, x)
  (s, v) = Two_Sum(v, x)
  (s, u) = Two_Sum(u, s)
  if (u == 0){
    u = s
    s = 0
  }
  if (v == 0){
    v = u
    u = s
    s = 0
  }
  return (s, u, v)

Triple_Add(a0, a1, a2, b0, b1, b2)
  (x[0], x[1], ..., x[5]) = merge_sort(a0, a1, a2, b0, b1, b2)
  u = 0
  v = 0
  k = 0
  i = 0
  while ((k < 4) && (i < 6)){
    (s, u, v) = Accumerate(u, v, x[i])
    if (s != 0){
      c[k++] = s
    }
    i++
  }
  if (k < 3){ c[k+1] = v }
  if (k < 4){ c[k] = u }
  return renormalize(c[0], c[1], c[2], c[3])
```

【図２７】

オーバーラップを除去するための3つのtwo__sum演算子の接続を示す図



【図２８】

符号の揃ったPriestの再正規化の例を示す図

```
t0  = 1200 0000 0000 0000
t1  =  -30 0000 0000 0000
t2  =    4 5600 0000 0000
t3  =                  -7

zz0≠0になるまで、t0,t1,t2をaccumulate

a0  = 1174 0000 0000 0000
z0  = 1174 0000 0000 0000
w0  = 0000 0000 0000 0000
zz0 =      5600 0000 0000

zz1≠0になるまで、w0,zz0,t3をaccumulate

a1  =      5599 0000 0000
z1  =      5600 0000 0000
w1  =         1 0000 0000
zz1 =                  -7

zz2≠0になるまで、w1,zz1をaccumulate

a2  =           9999 0000
z2  =         1 0000 0000
w2  =              1 0000
zz2 =                  -7

w2,zz2をaccumulate

a3  =                9993
z3  =                9993
w3  =                   0
zz3 =                   0
```

【図２９】

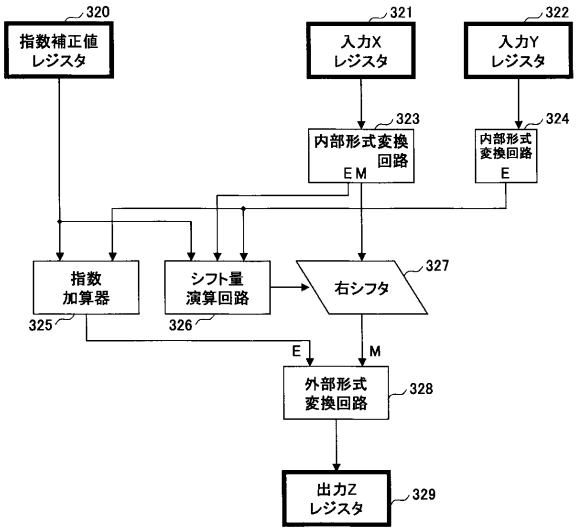
符号の揃ったPriestの再正規化された数の組から
強い正規化された数の組を得る計算の一例を示す図

```
a0 = 1.230 x 10^(15)          [1230] 0000 0000 0000
a1 = 4.050 x 10^(8)                  [4 050]0 0000
a2 = 6.000 x 10^(4)                        [6 000]0

b0 = a0                       [1230] 0000 0000 0000
   = 1.230 x 10^(15)
b1 = scale_next(a1,b0)              [0004] 0000 0000
   = 0.004 x 10^(11)
tmp = a1-b1+a2                          506 0000
b2 = scale_next(tmp,b1)             [0506] 0000
   = 0.506 x 10^(7)
```

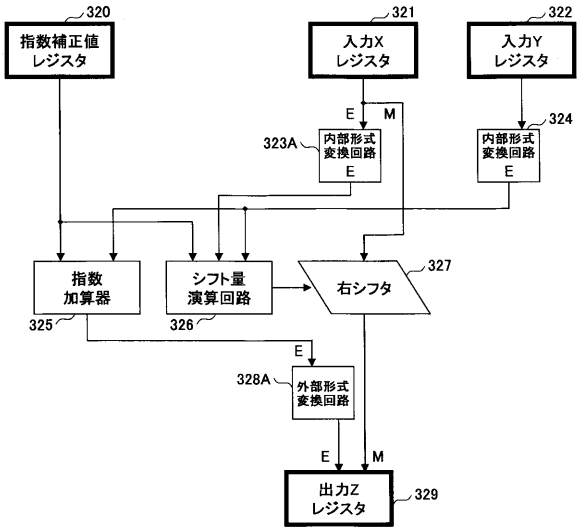
【図３０】

量子化のためのscale__next(X, Y)演算を実現する回路構成の一例を示す図



【図３１】

仮数部を内部形式に変換せずにscale__next演算を行う回路の構成の一例を示す図



【図３２】

Oracle－DatabaseにおけるNUMBER型の3種類の精度指定方法を示す表

書式	説明
NUMBER	精度指定なし
NUMBER(precision)	桁数による精度指定。precision は 1 以上 38 以下
NUMBER(precision, scale)	桁数とスケール指定による丸め位置指定。scale は小数点以下何位まで表すかを示し、-84 以上 127 以下の値を指定可能

【図３３】

四捨五入が発生する桁位置のみ5があるような数を生成するアルゴリズムの例を示す図

(a)

```
get_commma5(a0, digits=none)
  e = e(a0)
  r = 50 * 100^(e - 20)
  return r
```

(b)

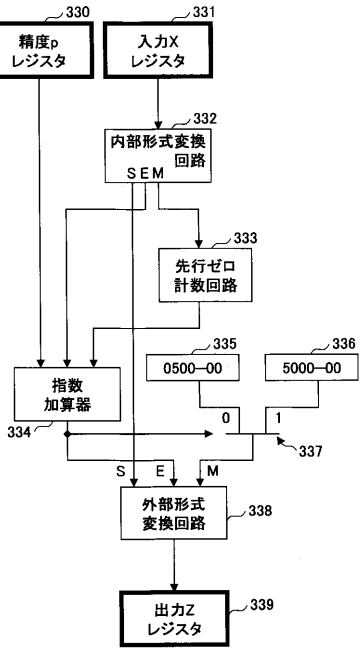
```
get_comma5(a0, digits=d)
  e' = e'(a0)
  r  = 5 * 10^(e - d)
  return r
```

(c)

```
get_comma5(scale=n)
  r = 5 * 10^-(n+1)
  return r
```

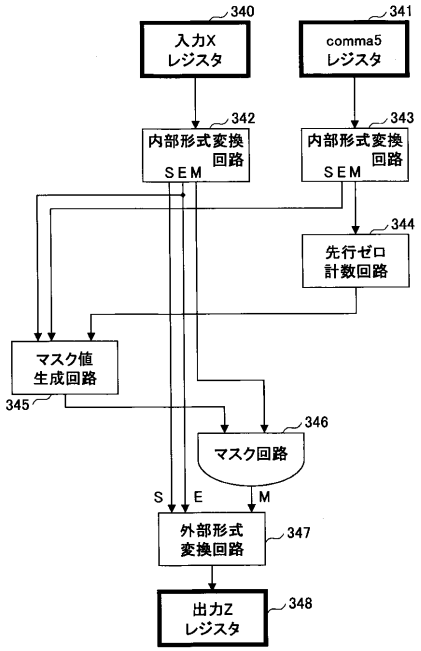
【図３４】

get__comma5演算を実現する回路構成の一例を示す図



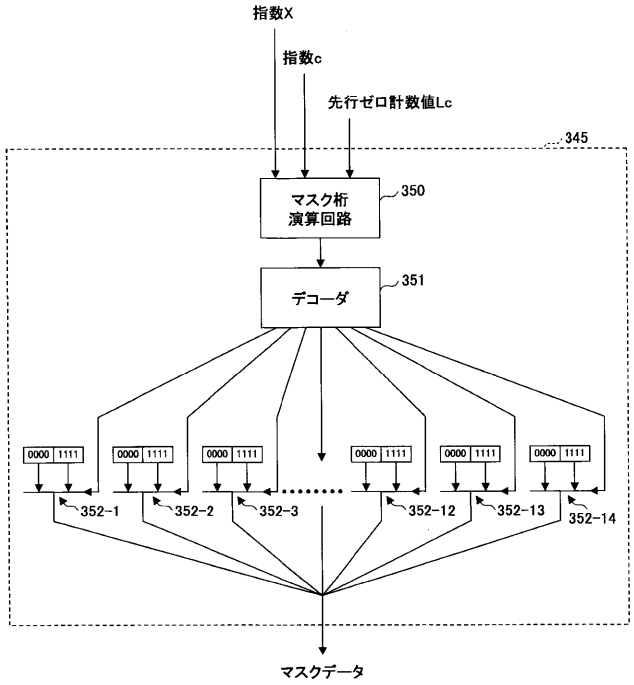
【図３５】

truncate演算を実現する回路構成の一例を示す図



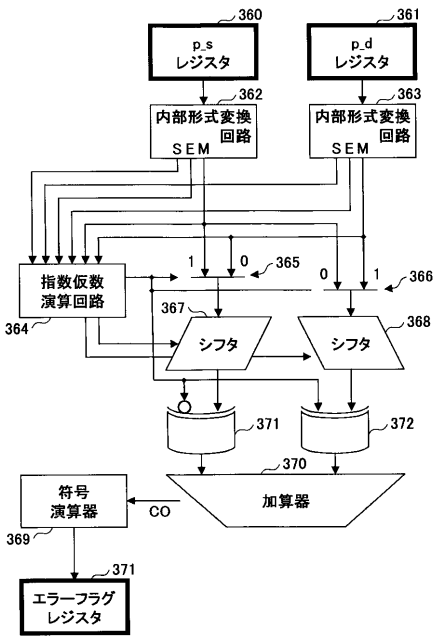
【図３６】

マスク値生成回路の構成の一例を示す図



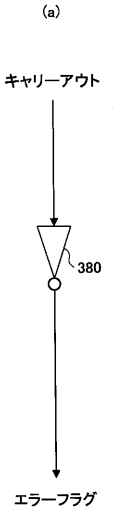
【図３７】

p__d及びp__sの大小比較エラー判定回路の構成の一例を示す図



【図３８】

符号演算器の回路構成の一例を示す図



(b)

大小関係	CO	エラーフラグ
|p_s| > |p_d|	1	0
|p_s| = |p_d|	1	0
|p_s| < |p_d|	0	1

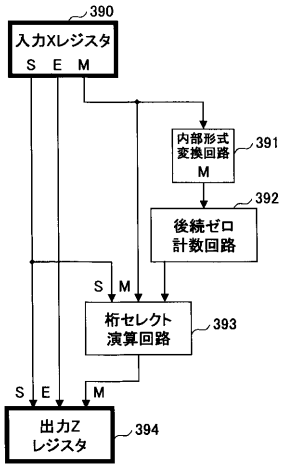
【図３９】

Oracle－numberの長さを求める演算の一例を示す図

```
get_oracle_num_length(a0, a1, a2)
  if (a2 != 0){
    l = 14 + min(7, length(a2))
  } else if (a1 != 0){
    l = 7 + length(a1)
  } else {
    l = length(a0)
  }
  return l
```

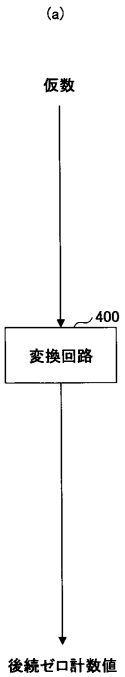
【図４０】

expand演算を実現する回路の構成の一例を示す図



【図４１】

後続ゼロ計数回路の構成の一例を示す図

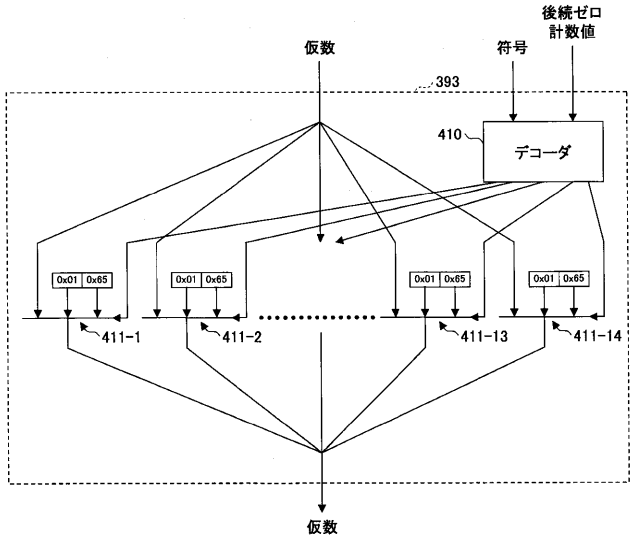


(b)

入力	出力
ｘｘｘｘｘｘｘｘｘｘｘｘｘｘｘ	００００
ｘｘｘｘｘｘｘｘｘｘｘｘｘｘ０	０００１
ｘｘｘｘｘｘｘｘｘｘｘｘｘ００	００１０
ｘｘｘｘｘｘｘｘｘｘｘｘ０００	００１１
ｘｘｘｘｘｘｘｘｘｘｘ００００	０１００
ｘｘｘｘｘｘｘｘｘｘ０００００	０１０１
ｘｘｘｘｘｘｘｘｘ０００００	０１１０
ｘｘｘｘｘｘｘｘ００００００	０１１１
ｘｘｘｘｘｘｘ０００００００	１０００
ｘｘｘｘｘｘ０００００００	１００１
ｘｘｘｘｘ０００００００	１０１０
ｘｘｘｘ０００００００	１０１１
ｘｘｘ０００００００	１１００
ｘｘ０００００００	１１０１
ｘ０００００００	１１１０
０００００００	１１１１

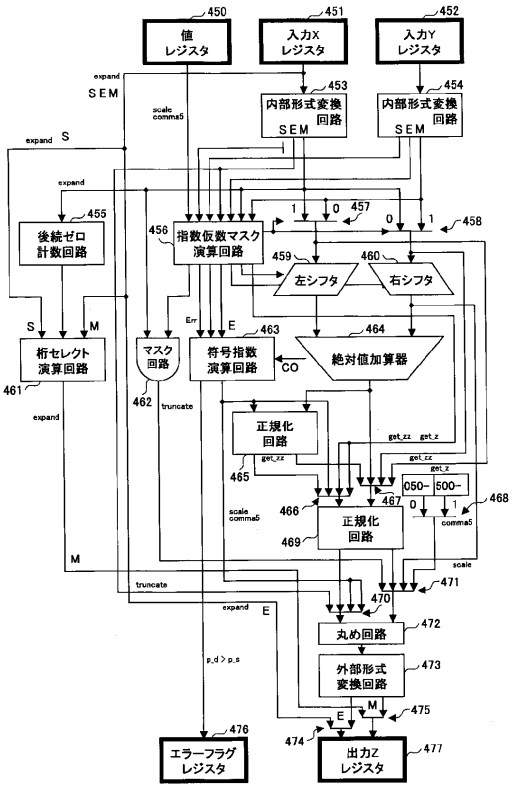
【図４２】

桁セレクト演算回路の構成の一例を示す図



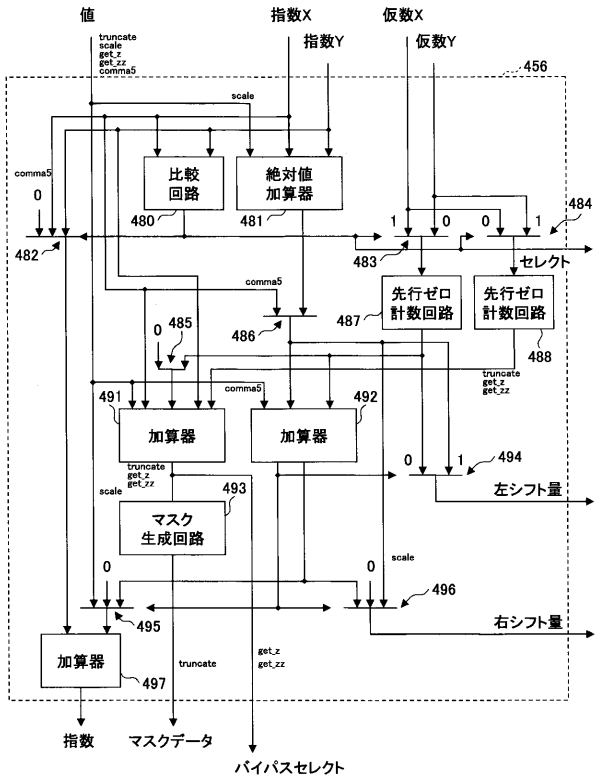
【図４３】

固定精度浮動小数点加減算器の構成の一例を示す図



【図４４】

指数仮数マスク演算回路の構成の一例を示す図



【手続補正書】
【提出日】平成24年1月10日(2012.1.10)
【手続補正１】
【補正対象書類名】明細書
【補正対象項目名】０１４９
【補正方法】変更
【補正の内容】
【０１４９】
以下に、ｔｒｉｐｌｅ－ｏｒａｃｌｅｎｕｍ６４形式に対する丸め処理を考える。ここでは、丸め処理対象であるｔｒｉｐｌｅ－ｏｒａｃｌｅｎｕｍ６４形式の数の組（ａ０，ａ１，ａ２）は再正規化されていることを前提とする。なおこの前提条件は、符号の揃ったＰｒｉｅｓｔの再正規化及び強い再正規化の何れであってもよい。

フロントページの続き

(72)発明者　北村　健一
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内