誤差情報を含む浮動小数点表現と
これを用いた数値演算論理

シグナル・プロセス・ロジック株式会社
瀬尾雄三

発表内容

この研究の背景と目的

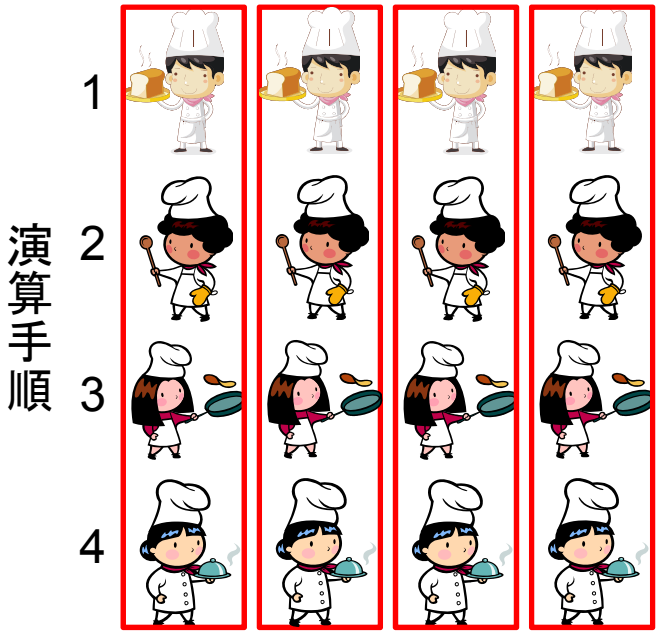
誤差情報を含む浮動小数点表現

数値演算論理設計への応用

まとめ

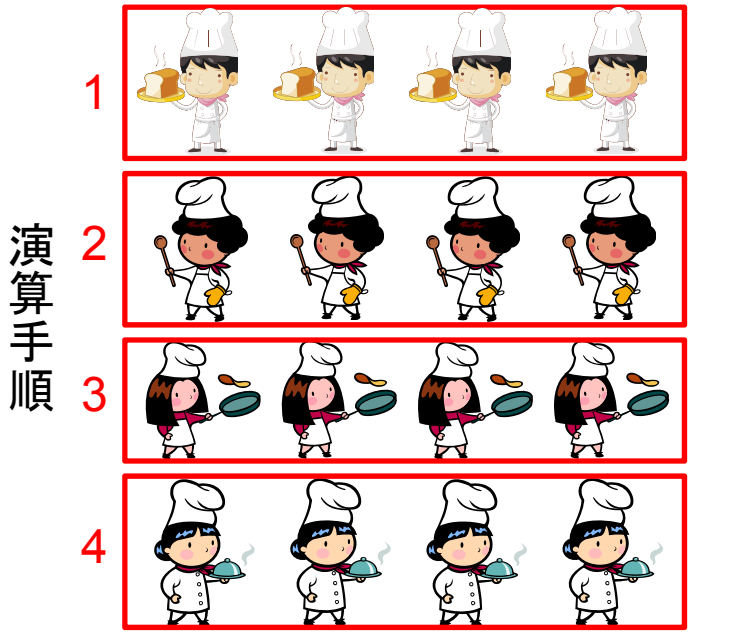
背景：高速演算処理の二つのアプローチ

マルチCPU：データ毎に
異なるCPUで処理する



多機能演算器（CPU）
複雑だがすべてが同一

パイプライン：演算手順毎に
異なる演算器で処理する



単機能演算器
簡素だが各々異なる

高速処理を安価に実現!

設計が大変!

目的：パイプライン用演算器の自動設計

演算器を最小規模で構成したい

 ビット幅の最適化が効果的

入力ビット幅 信号処理なら ADC分解能	十進 桁数
～10	3
～13	4
～16	5
～20	6
～23	7

CPUでは仮数部53 bit幅*の演算が一般的
必要なbit幅はこの半分以下の場合が多い

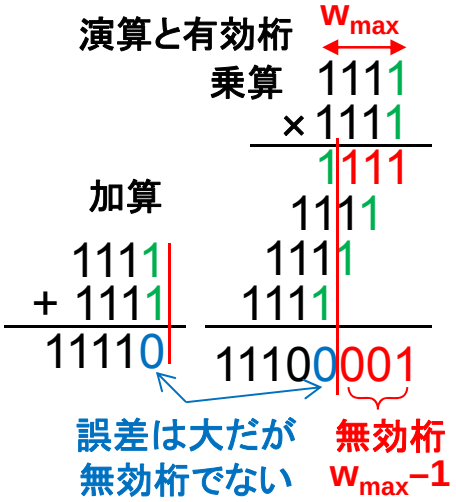
* IEEE754倍精度浮動小数点形式

加減算の論理規模∝ビット幅
乗除算の論理規模∝ビット幅^2

必要な桁のみを演算することで論理規模を最小化

- 誤差を含まない数：値の変動範囲でビット幅を規定
- 誤差を含む数：有効桁のみの処理で更に規模縮小

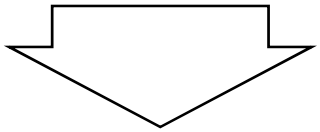
有効桁を扱うための数値型の条件



- 有効桁を扱うには浮動小数点表現が必要
- 誤差は大きいが無効桁ではない桁の存在
  ⇒　桁数による精度表示だけでは不十分
- 誤差キャンセルの問題

(1) c = a + b
(2) d = c – b = a

bが支配的の場合、(1)でaの情報が失われる
検出にはcに含まれるb起因の誤差量が必要

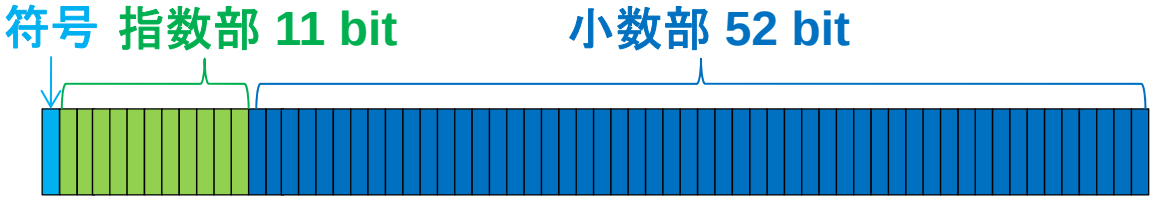


要因別誤差情報を含む浮動小数点表現

IEEE754浮動小数点表現

- 倍精度形式は符号、指数部11 bitおよび小数部52 bitで構成
  - 指数部－1023が指数値、指数部の2047は異常を意味する
  - 仮数部最上位は通常省略し、指数部0の場合のみ小数部に含む

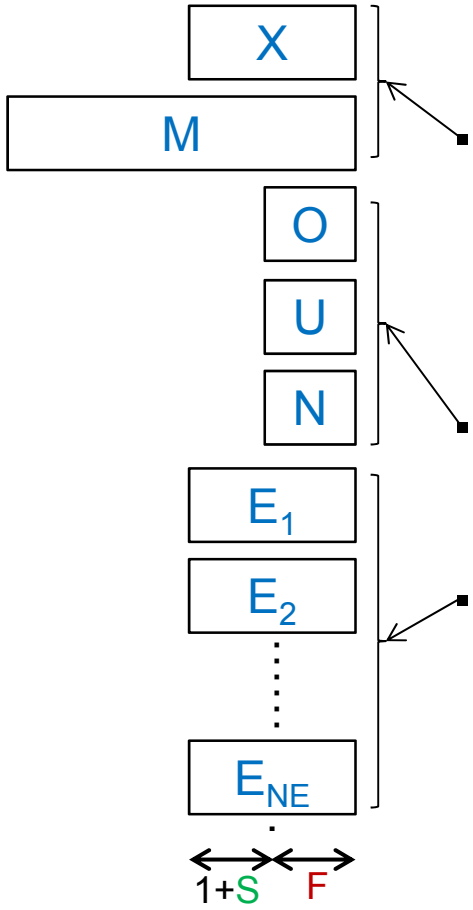
  } 情報圧縮



- 情報圧縮：記憶エリアは節約、処理する論理は複雑
- CPUは演算器が一つで、記憶領域が多数なので合理的
- パイプラインは演算器が多いため簡素な処理が好ましい

IEEE754とは異なる浮動小数点表現を使う

提案：誤差情報を含む浮動小数点表現



S: 無効桁数
F：小数部桁数

仮数部、指数部とも任意ビット長の整数とする

- 仮数部M、指数部X、値 = M 2^X
- 数値型は最大値(Xmax, Mmax)と最小値(Xmin, Mmin)で規定
- 最小値が負の場合は符号付。負数は二の補数で表す

異常表示には専用の信号線を設ける

- 数値化不能：N、正の無限大:O、負の無限大:U

要因毎の誤差指標を配列で与える：E[NE]

- Mの無効桁数Sを規定することで、指数部Xは共用される
- 微小誤差表現のため、仮数部にF桁の小数部を持たせる
- 整数部桁数は、無効桁数S+1符号ビット
  + 通常はS=0として有効桁のみを伝達する
  + 誤差のキャンセルが生じる場合はSを増加して対処する

誤差と有効桁の管理

- 誤差指標には誤差の最大値を用いる（以下これを「誤差」と呼ぶ）
  - 誤差の最大値：量子化誤差（仮数部LSB±1/2）を1/2と表示すること
    厳密な誤差評価には標準偏差への換算が必要
- 誤差の値 $e_k = E_k \, 2^{X-F}$
  - $(e_1, e_2, \ldots e_{NE}) = \boldsymbol{e}$ を誤差ベクトルと呼ぶ
- 誤差分散は $\boldsymbol{e}^2 \equiv \sum_{k=1}^{n} e_k^{\,2} \equiv \sum_{k=1}^{n} (E_k \, 2^{-F})^2 \, 4^X$ で与えられる

  誤差分散の仮数部
- 誤差分散による演算結果の正規化
  - 誤差分散の仮数部が4^{S-1}~4^Sとなるよう指数部Xを定める（S：無効桁数）
- 誤差分散仮数部が目標範囲上限以上の場合
  - 演算結果の指数部をインクリメントし、値と誤差の仮数部を1 bit丸める
    + 操作1 bitで誤差分散の仮数部は1/4となる
- 誤差分散仮数部が下限未満なら誤差キャンセルが発生
  - 入力信号に含まれる無効桁を増加させて対処（詳細は後述）

演算結果に含まれる誤差

a, b：変数、c：実定数、$\boldsymbol{e}$：誤差ベクトル

▪ 加算：	$r = a + b$,	$\boldsymbol{e}_r = \boldsymbol{e}_a + \boldsymbol{e}_b$	$r' = a' + b'$
▪ 減算：	$r = a - b$,	$\boldsymbol{e}_r = \boldsymbol{e}_a - \boldsymbol{e}_b$	$r' = a' - b'$
▪ 定数倍：	$r = c\,a$,	$\boldsymbol{e}_r = c\,\boldsymbol{e}_a$	$r' = c\,a'$
▪ 乗算：	$r = a\,b$,	$\boldsymbol{e}_r = b\,\boldsymbol{e}_a + a\,\boldsymbol{e}_b$	$r' = b\,a' + a\,b'$
▪ 除算：	$r = a / b$,	$\boldsymbol{e}_r = \boldsymbol{e}_a / b - a\,\boldsymbol{e}_b / b^2$	$r' = a'/b - a\,b'/b^2$
▪ 平方根：	$r = \sqrt{a}$,	$\boldsymbol{e}_r = \boldsymbol{e}_a / 2\sqrt{a}$	$r' = a' / 2\sqrt{a}$
▪ 定数乗：	$r = a^c$,	$\boldsymbol{e}_r = c\,a^{c-1}\,\boldsymbol{e}_a$	$r' = c\,a^{c-1}\,a'$

a, b, rを関数とみなして$\boldsymbol{e}_a \rightarrow a'$、$\boldsymbol{e}_b \rightarrow b'$、$\boldsymbol{e}_r \rightarrow r'$と書き換えれば微分公式に対応

- 誤差は入力の値に依存 ⇒ 信号を処理する際に決まる
  - 誤差情報を用いるなら、誤差を演算する論理を実装する
  - 誤差分散の値に応じて無効桁の丸め処理を行う論理を形成
- 信号線幅は、事前のシミュレーションにより決定する

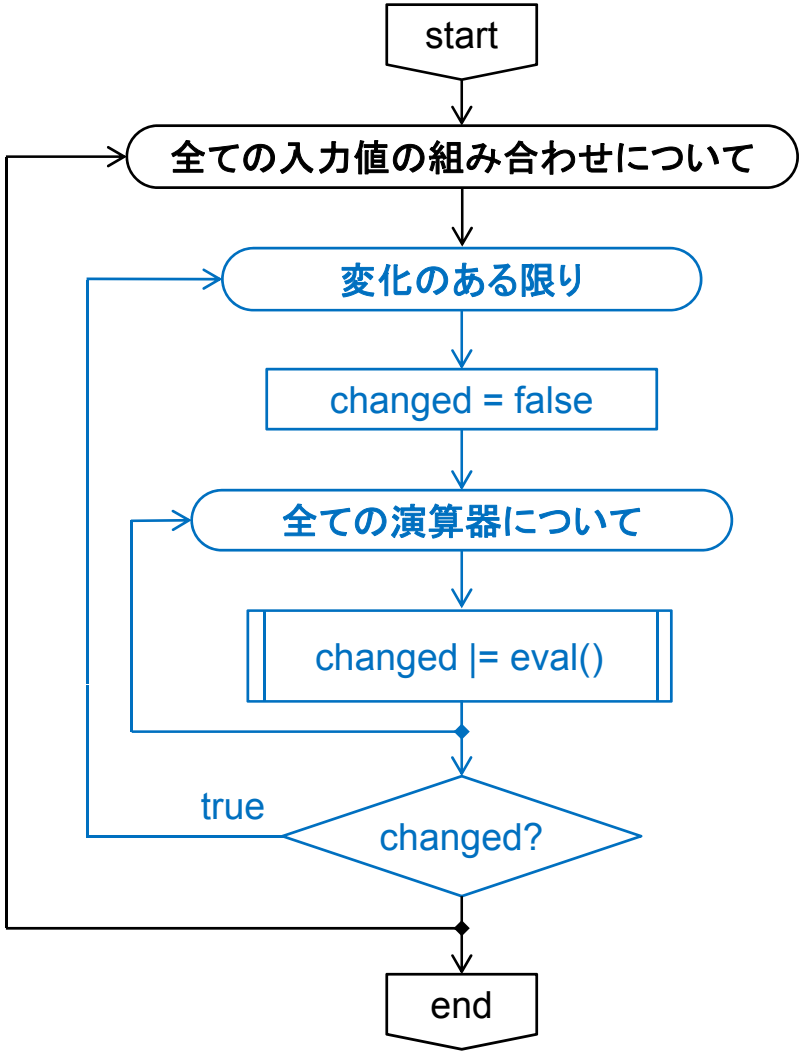
ソフトウエアによる実現

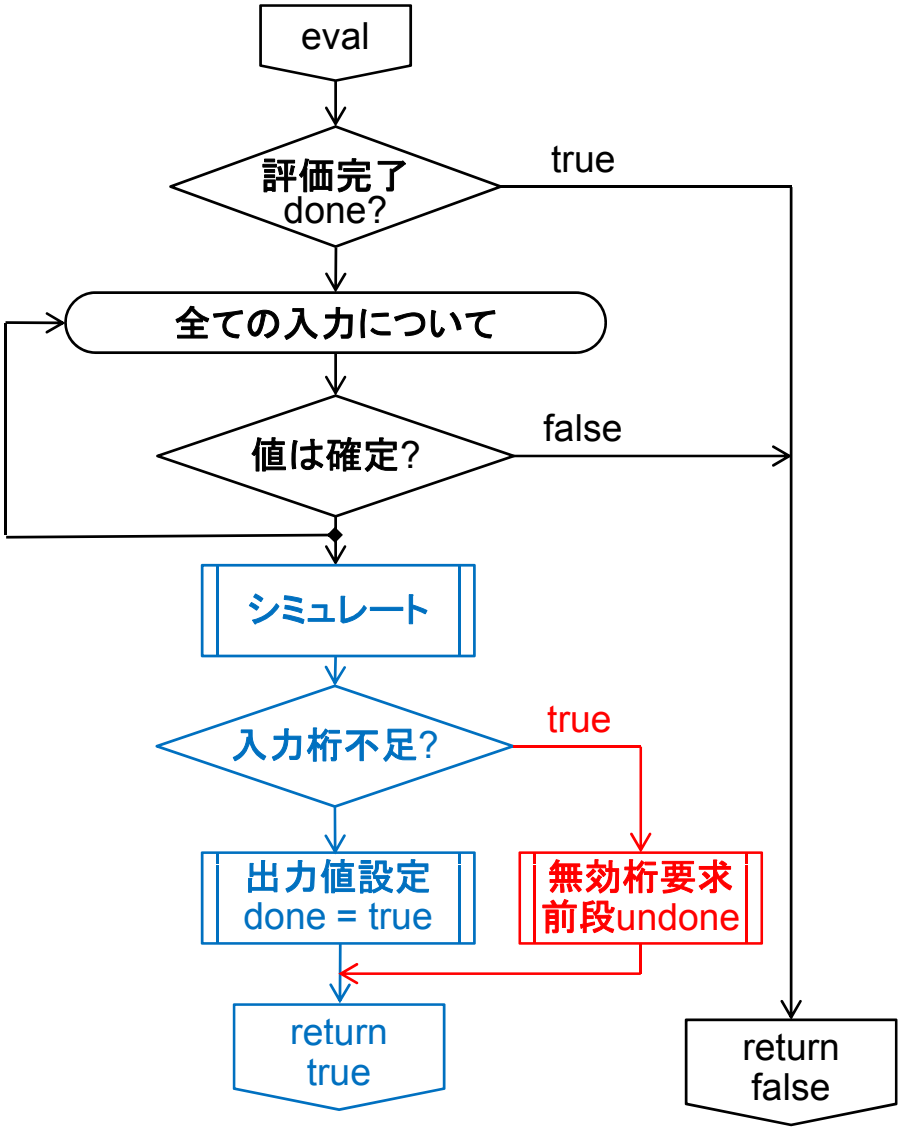
- 誤差情報を含む浮動小数点数クラス
  - 値rと誤差要素配列eからなる
  - 演算子と各種関数は独自に定義
  - シミュレーションの際、各信号線の状態を表示
  - 一般の科学技術計算への応用も

```
public ref class Real
{
public:
    double r;
    array<double>^ e;
    long long m, x;
    int s;
    double ve(void); // 誤差分散
    Real^ operator+(Real^ src);
    ……
};
```

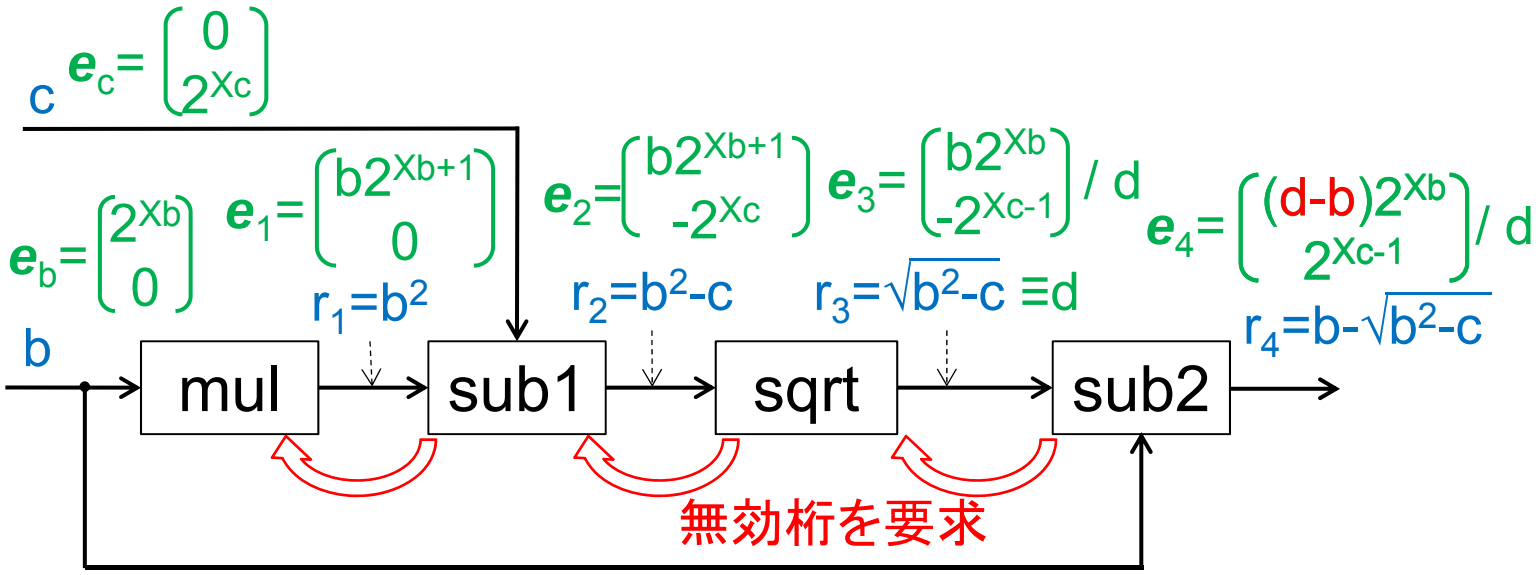
- シミュレーション
  - 外部入力の全ての（または代表的な）値の組み合わせでシミュレート
  - 各入力値に対して、全演算器の出力する値と誤差ベクトルを算出する
  - 値と誤差分散から仮数部mと指数部xを決定する（$\boldsymbol{e}^2/4^x=4^{s-1}\sim4^s$）
  - 入力の仮数部bit幅が不足した場合誤差キャンセルへの対処が必要
    - + 不足したbit幅に応じて、入力信号線の無効桁数sを増加
    - + これを出力信号線とする演算器を再評価し、
    - + その演算器出力を使用している演算器も再評価する

シミュレーションのフロー





例: $x = b - \sqrt{b^2 - c}$の演算論理



- e_4 計算部分でdとbが同符号の場合に誤差がキャンセルする
- 前段の出力に無効桁を含めるよう要求する。前段も更に前段に要求

- 演算アルゴリズムの改善による誤差キャンセルの解消
  - bが正の場合は$b+\sqrt{b^2-c}$ を分子分母に乗じた$c / (b+\sqrt{b^2-c})$ を用いるのが良い
  - 数式処理で対応する可能性もあるが、当面は、警告を出力して対処を促す

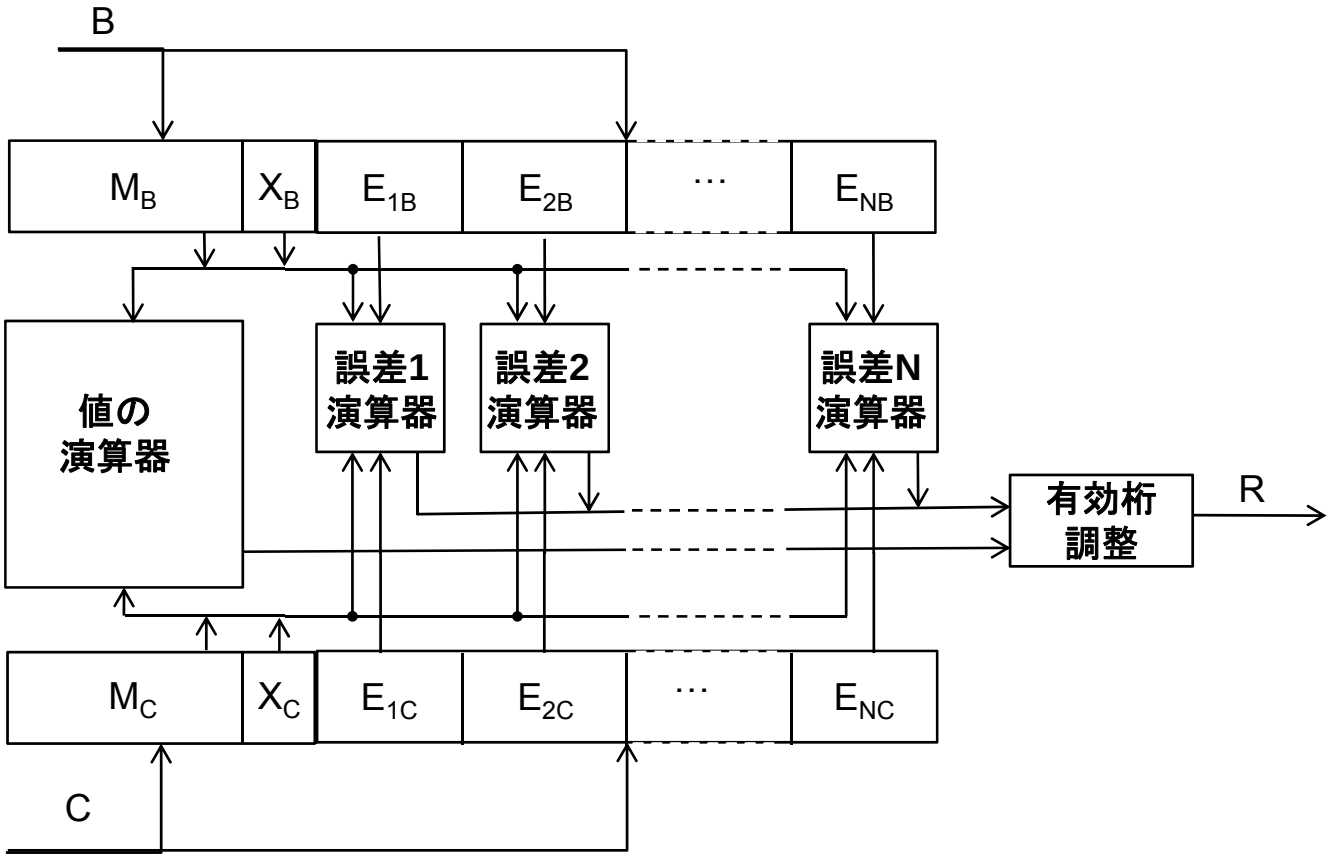
誤差情報の利用と最大伝送モード

- 誤差情報の利用
  - 演算結果の信頼性保証：高信頼性が要求される分野で有益
  - ゼロでないことの統計的判定：IEEE754浮動小数点形式では不可能
  - 誤差の大きさに基づく処理の選択
  - … などなど（これらはclass Realを用いる演算においても利用できる）

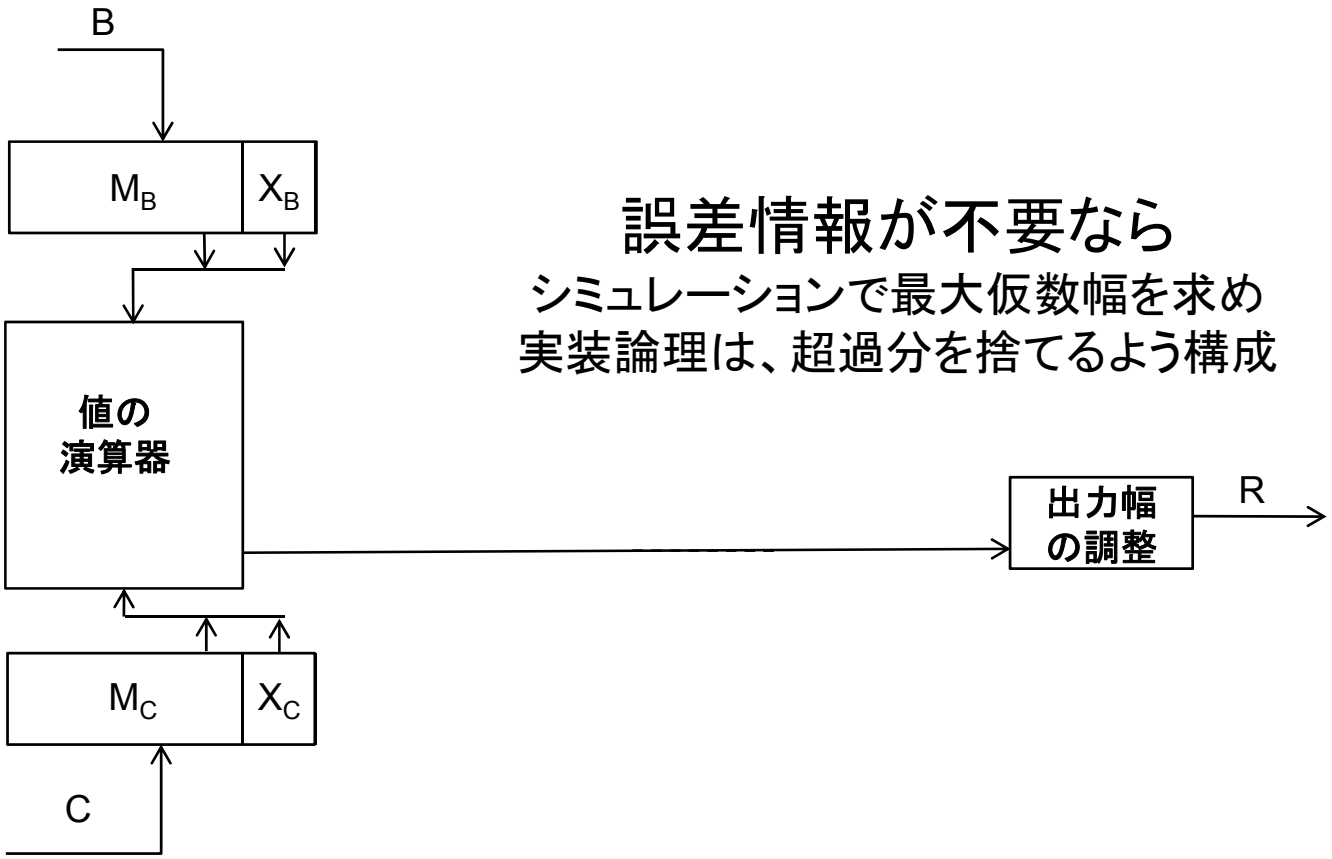
- **誤差情報を利用しない場合（最大伝送モード）**
  - 演算実行中に誤差情報を利用しないなら、誤差演算論理は省略可能
  - 仮数部信号線幅は、シミュレーションで必要な最大幅に設定
  - 演算論理は、最大幅を超える際に下位を丸めるよう構成

誤差情報が必要なら実装、不要なら非実装

演算論理回路


B
M_B
X_B
E_{1B}
E_{2B}
…
E_{NB}
値の
演算器
誤差1
演算器
誤差2
演算器
誤差N
演算器
有効桁
調整
R
M_C
X_C
E_{1C}
E_{2C}
…
E_{NC}
C

演算論理回路



誤差情報が不要なら

シミュレーションで最大仮数幅を求め

実装論理は、超過分を捨てるよう構成

まとめ

- パイプライン演算に適した浮動小数点形式を提案した
- 要因別の誤差情報により有効桁が正確に管理される
- 要因別誤差情報により誤差キャンセルも検出される
- これらにより必要最小規模の演算論理設計が可能となる
- 誤差情報は演算内での制御や結果の信頼性保証に有益
- 誤差の演算論理を実装しない、簡素な論理も構成可能
- 今後、本手法を用いた設計支援システムを実用化したい
- 本手法は、一般の科学技術計算にも応用可能と思われる