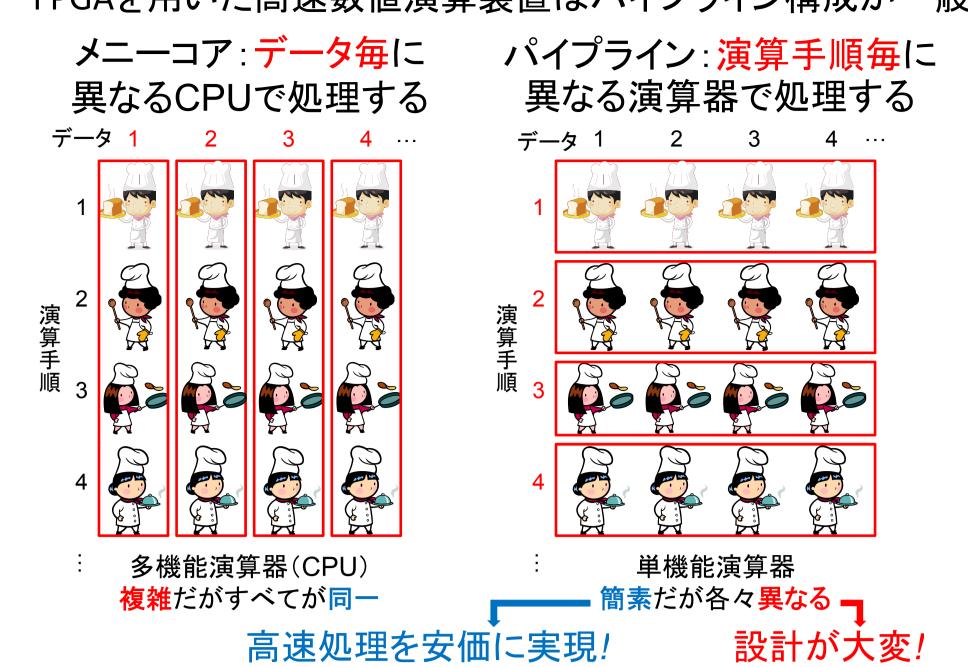
要因別誤差追跡に基づく安全な数値演算論理の自動設計

シグナル・プロセス・ロジック株式会社 瀬尾雄三

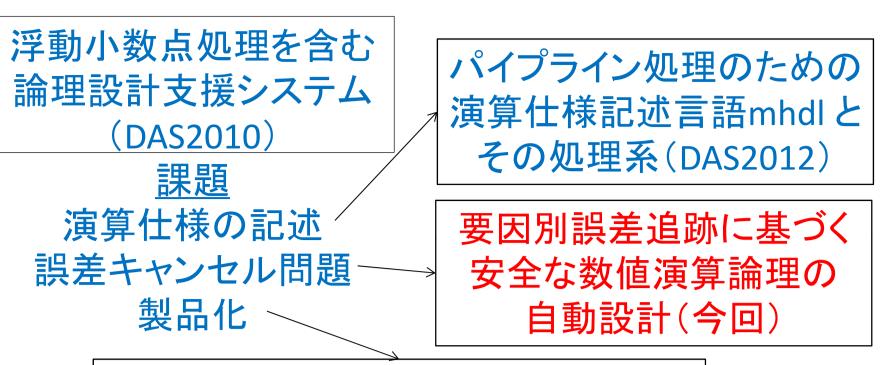
本研究の目的:パイプライン演算器の自動設計

• FPGAを用いた高速数値演算装置はパイプライン構成が一般的



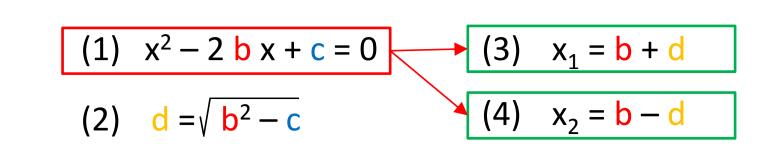
数値演算論理の最適設計:検討の流れ

- 有効桁に着目して、必要最小のビット幅で演算論理を形成 する手法をDAS2010で発表:残された課題は、
- 演算仕様の記述: DAS2012で演算仕様記述言語を発表
- 誤差キャンセルの問題: 今回発表
- 論理設計支援システムとしての製品化:評価版を公開中



CodeSqueezer 評価版を公開中 (URL: http://signal-process-logic.com)

誤差キャンセルはなぜ問題か



- (1)式の二つの解x₁およびx₂は(2)~(4)式で与えられる
- (3)、(4)のいずれかでbに含まれる誤差がキャンセルされる (bの符号に依存)
- bの誤差がcに比べて大きい場合
- (2)式:bの誤差がdの有効桁を決定する
- (3, 4)式:bの誤差がキャンセルされるが cに含まれる情報は既に失われている
- 対処:後段で誤差がキャンセルする場合、dに無効桁を 含め、cに含まれる情報を後段に伝達する

要因別誤差追跡の考え方

- (1) $x^2 2bx + c = 0$ (3) $x_1 = b + d$
- (2) $d = \sqrt{b^2 c}$
- (4) $x_2 = b d$
- (3,4)式でbの誤差がキャンセルされることを検出するには、 変数dに含まれるb起因の誤差の大きさを知る必要がある
- dの誤差要因には、変数bと変数cがある
- これらを識別可能とするため、要因別の誤差情報を扱う - 基本演算を定義しておけばdoubleと同様に扱うことが可能
 - public ref class Real public: double r; array<double>^ e;

基本演算の定義

Real^ operator+(Real^ src); Real^ operator— (Real^ src); Real^ operator*(Real^ src); Real^ operator/(Real^ src); Real^ sqrt(Real^ src);

演算結果に含まれる誤差

- 各種演算の結果に含まれる誤差は下式で計算される これらは、誤差の高次項を無視して導き出されている
- 0どうしの乗算においては誤差の高次項が無視できない → **δ**を追加:誤差の独立性より、異なる誤差要因の積は無視

a, b, rは値e, e, e, は誤差、cはdouble型の値

- (5) $r = a + b : e_r = e_a + e_b$
- (6) $r = a b : e_r = e_a e_b$ (7) $r = c * a : e_r = c e_a$
- (8) $r = a * b : e_r = b e_a + a e_b + \delta$
- $\delta = (e_{a1} e_{b1}, e_{a2} e_{b2}, ...)$ (9) $r = a / b : e_r = e_a / b - a e_b / b^2$
- (10) $r = \sqrt{a} : e_r = e_a / (2\sqrt{a})$
- (11) $r = a^c : e_r = c a^{c-1}e_a$

誤差とは

- 一般には扱う値と真値の差:計測誤差、ノイズ等を含む

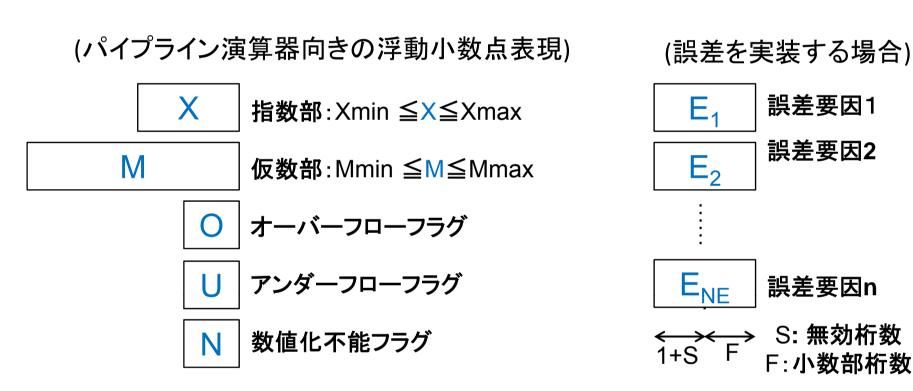
誤差と有効桁

- 本論では量子化誤差で誤差を表現する: ±½LSB
- 個々の入力に対して誤差を設定することも可能
- 誤差: 誤差分散($e^2 \equiv \sum e_k^2$)の平方根 ϵ を誤差とみなす
- 重みが誤差以下の桁よりも下位は無効桁



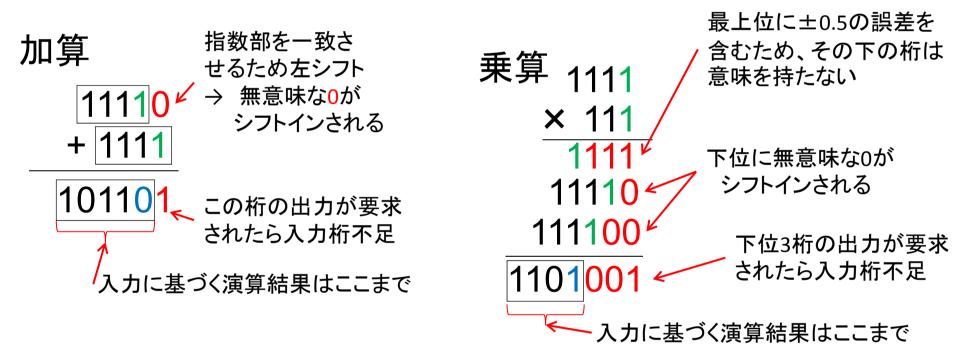
浮動小数点数の実装と無効桁数の制御

- 値を仮数部Mと指数部Xを用いてM×2^xで表す
- 量子化誤差=0.5×2×より、誤差εが2×-1~2×なら無効桁数は0
- 無効桁数をS bitとするにはεを2^{X+S-1}~2^{X+S}とするようXを定める 指数部Xは $X = log_2 \epsilon - S$ で計算される(小数部は切捨て) (実際には、 $X = \log_a \varepsilon^2 - S$ で計算。 $\log_a O$ 整数部はビット幅/2)



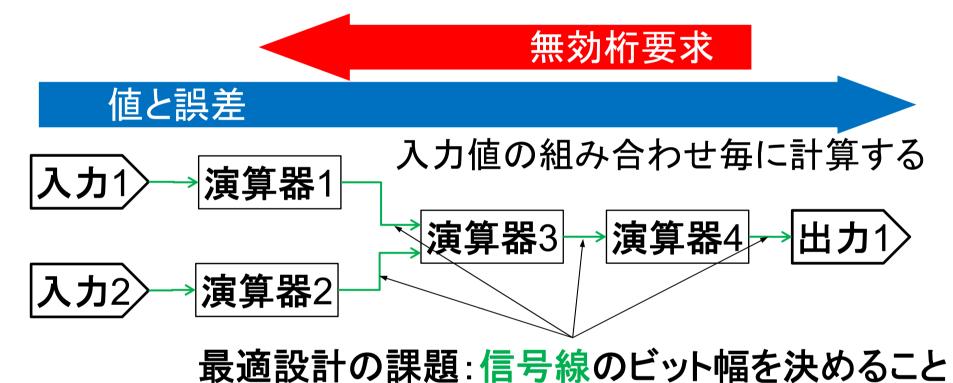
誤差キャンセルの検出

- 誤差のキャンセルは入力値の特定の組み合わせで発生 → 事前のシミュレーションでチェックする
- 無効桁の調整:出力の指数部X設定→仮数部Mも決まる
- 出力仮数部への入力仮数部のLSB以下の影響をチェック → 影響するなら誤差キャンセルが問題を生じている
- 演算毎に誤差キャンセルの検出手法は異なる



誤差キャンセルへの対応

- 入力側から値と誤差を計算(入力値の組み合わせに対し)
- 入力仮数部の桁不足が検出されたら前段出力の無効桁を 増加させる(後段から前段へと無効桁要求が伝播する)
- 演算結果の丸めを経ている場合のみ無効桁の増加が可能



信号線のデータ構造

- 各入力値の組に対する値と誤差(シミュレーションで使用) - 値と誤差ベクトル(Real型)
 - 信号線に表れた数値の仮数部と指数部の値
 - 信号線に要求される無効桁数
 - 無効桁増加可能フラグ: 丸めが行われていることを示す
- 実装する信号線の属性(全ての入力値に対応すべく設定) - 仮数部と指数部の最大値と最小値
- 要求無効桁数と有効桁数の最大値(誤差実装時に使用)
- 異常信号線(オーバーフロー、アンダーフロー、数値化不能) - 異常信号線の有無と、存在する場合はその属性

演算過程の一例

• $x^2 - 2bx + c = 0$ の一解 $-\sqrt{b^2-c}$ を演算するフロー

e₄の1行目d-bで誤差のキャンセルが発生する

- この結果、順次前段に無効桁要求が送られる

入力値が決定している初段より順に値が設定される

・ 前段出力に無効桁を要求→前段の入力も桁不足が発生

- 接続関係の定義
 - 信号源となる演算器

2^{Xc-1}

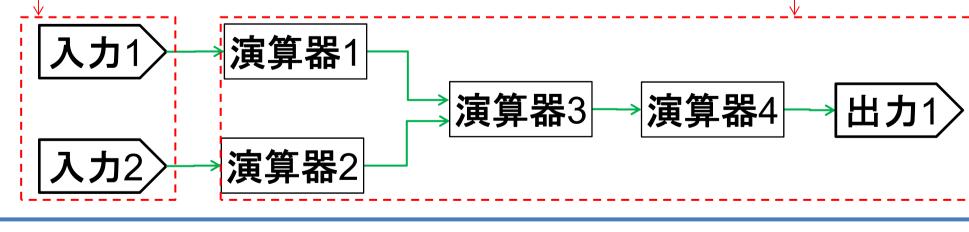
まとめと

今後の課題

- 信号の行き先となる演算器のリスト

演算仕様のモデル化

- 演算器のデータ構造
 - 入力に接続される信号線のリスト
 - 出力に接続される信号線のリスト
 - 値を評価する関数(eval)へのポインタ
 - 演算器をHDL記述に変換する関数へのポインタ
- 作業完了(done)フラグ
- 演算仕様のモデル化
 - 外部入力信号線のリストと各信号線の属性 - モデルに含まれる演算器と出力のリスト
 - 演算器入出力と信号線の接続関係



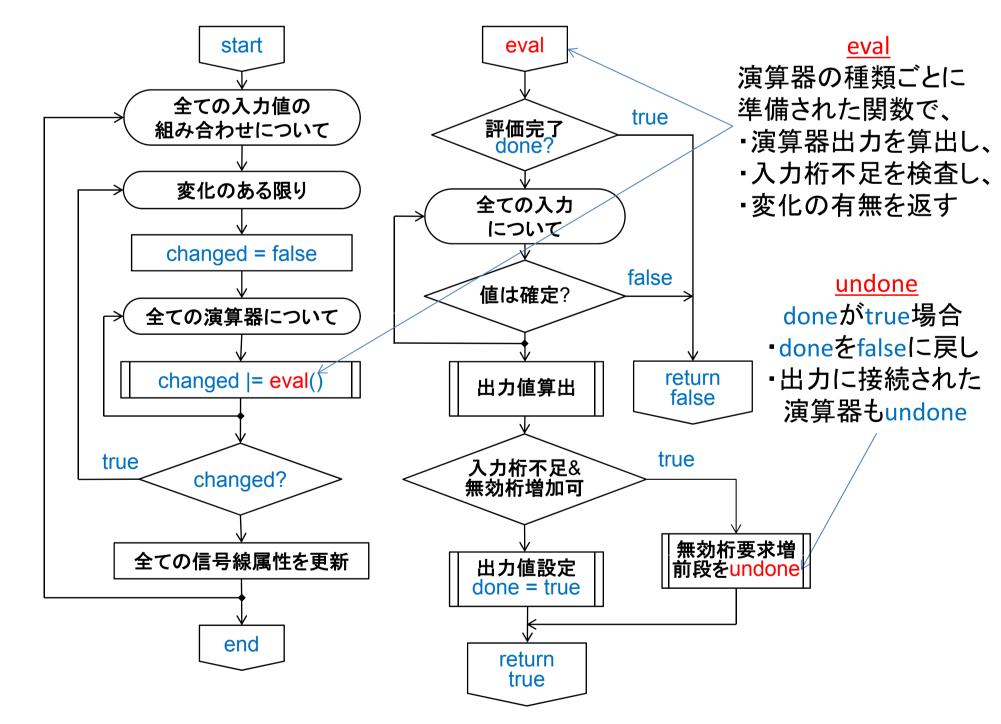
必要無効桁数の計算例(最終段)

- r3 = √b²-c に要求される無効桁数は下表の通りとなった
- bの仮数部M_bは1,000に、cの仮数部M_cは1,000,000に固定
- 指数部(X_b, X_c)を各々0から10まで振ってシミュレートした



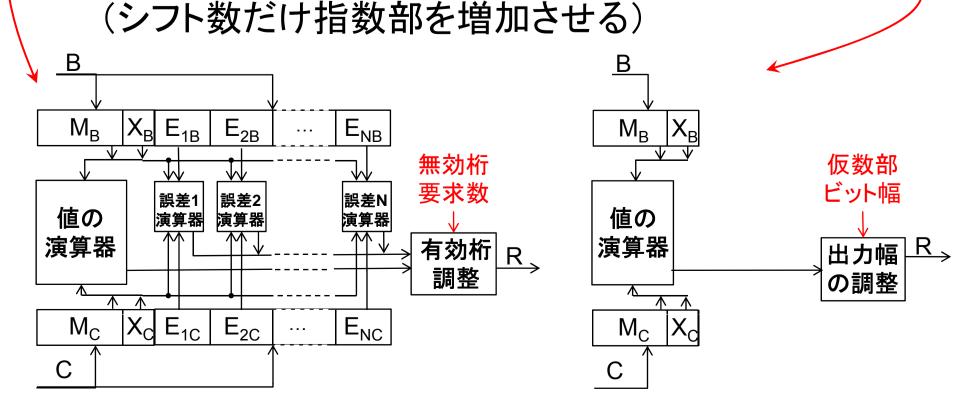
• b²とcの大小関係で無効桁必要数が与えられている

シミュレーション:処理の流れ



誤差情報の利用と最大伝送モード

- 実装演算論理で誤差情報を利用する場合
- 誤差を表す信号線と誤差の演算論理を実装する
- 演算器は出力に要求される無効桁数を出力する
- 実装演算論理には誤差情報が不要の場合
- 誤差に係わる信号線と演算論理は実装しない - 出力の仮数部を信号線幅に収まるよう右シフトする



誤差キャンセルの機械的な検出・対応手法を提案した

- 有効桁に着目した数値演算論理の最適化に有用

sub2

 $r_4=b-\sqrt{b^2-c}$

- 一般の科学技術計算への応用も可能か
- 手法に関する課題:よりきめ細かな制御手法を検討する - 特異点に対する処理の適正化:ゼロの平方根など
- 今後は、本手法を応用した数値演算 論理設計支援ツールの開発を急ぎたい

無効桁を要求

- 無効桁数の上限設定:桁不足による新たな誤差の導入