B-001

URR 浮動小数点数のためのレイテンシを短縮した 64 ビット FPU の実装

Implementation of a 64bit FPU with Reduced Latency for URR Floating-Point Arithmetic

大山　光男 †
Mitsuo Ooyama

1．はじめに

可変長指数部を持つ URR[1]の実用化を阻害する要因の一つが，演算に手間がかかることであり，スループットの点でも不利と見なされている．そこで筆者らは，IEEE 標準の浮動小数点数を演算する FPU に迫るスループットを目標に，指数と仮数の分離，結合のステージを演算パイプから削除した方式を提案している[2]．今回，この方式に基づく URR，拡張 URR[3]のための 64bitFPU を新たに設計し，FPGA を用いて実現した．平方根演算を含む主要な演算命令を実装したことで，いくつかの実用的なプログラムの実行が可能となった．そこで今回は，Graeffe 法による 4 次方程式の根の計算プログラムを実行させ，そのスループット，および計算結果に含まれる相対誤差の評価を行った．

2．URR 浮動小数点数の概要

図 1 に URR のデータフォーマットを示す．

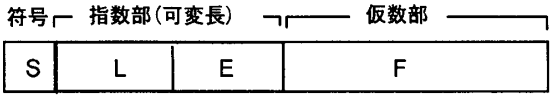

図 1．　URR 浮動小数点数のデータフォーマット

今，実数xを，$x = 2^e \times f$ と表わす．
指数 e は整数，仮数 f の正規化条件は

$1 \leqq f < 2$　　($f \geqq 0$)
$-2 \leqq f < -1$　　($f < 0$)

である．図 1 において，符号ビット S は仮数 f の符号，仮数部 F は，仮数 f の小数点以下のビットが上位から順に入る．指数部 L，E は，指数 e が m+2 ビットで

$e = Se\ \overline{Se}\ e_{m-1} \cdots e_0$　　(Seは符号)

と表され，$x \geqq 0$ のときは

$\underbrace{1 \cdots 10}_{m+2\text{個}}\ e_{m-1} \cdots e_0$　，($e \geqq 0$)

$\underbrace{0 \cdots 01}_{m+2\text{個}}\ e_{m-1} \cdots e_0$　，($e < 0$)

となる．ただし，e が -2,-1,0,1 の各場合は E は空であり，L は，それぞれ，001, 01, 10, 100 である．$x < 0$ の場合は，上記 L,E の各ビットを 1/0 反転して指数部とする．

拡張 URR は，URR が 2 重指数分割を $\pm 2^{\pm 2^m}$ で行うのに対して，$\pm p^{\pm q^m}$ で行う．指数部の構成は複雑になるが，指数が大きい領域で指数部の長さが大幅に短縮される．本 FPU では，表現効率がよいとされる，p=4，q=16 の場合を扱う[3]．

† 倉敷芸術科学大学　コンピュータ情報学科

3．FPU の設計と実装

表 1 に示すように，64bit の URR，あるいは拡張 URR から，64bit の仮数と 32bit の指数を分離して浮動小数点レジスタに格納し，演算命令の演算対象とする．

図 2 に示すように，演算パイプは指数と仮数の分離，結合ステージを含まず，レイテンシの不利は解消される．分離は，ロード命令に続く独立した 1 ステージで行う．結合は，演算命令の浮動小数点レジスタ更新に続く独立した 1 ステージで行う．バッファレジスタは，浮動小数点レジスタの内容のコピーを URR で保持し，ストア命令に先行して結合した結果を格納する．

パイプラインにおけるステージの数は，メモリアクセス，分離と結合，加減算，正規化，整数命令実行が各 1，乗算が 2 である．乗算では，FPGA が多数内蔵する乗算器を用いて高速乗算器を構成，利用する．除算と平方根は非パイプライン処理であるが，高速乗算器を活用，

表 1．FPU の仕様

項　目	仕　　様
データ	[URR(64bit) / 拡張 URR(64bit)] ⇄ 分離 / 結合 ⇄ [指数(32bit) / 仮数(64bit)]
浮動小数点レジスタ	浮動小数点レジスタ: (32bit+64bit)×32 本 同バッファレジスタ: 64bit×32 本
命令	浮動小数点：　加算，減算，乗算，除算，積和演算，平方根，ロード／ストア，レジスタコピー，他． 整数：　フロー制御命令，他．

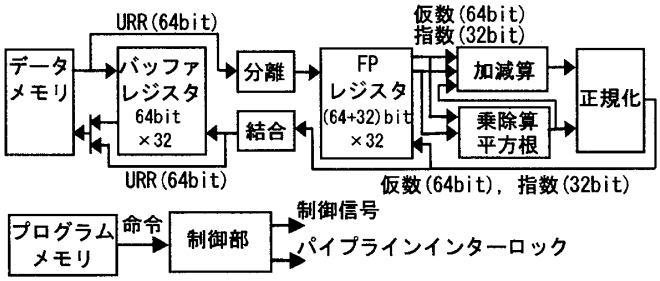

図 2．FPU の構成

表 2．命令実行レイテンシとスループット

命　令	レイテンシ	スループット
ロード／ストア	1 ／1	1
加算，減算，絶対値，レジスタコピー	2	1
乗算／積和	3 ／ 4	1
除算／平方根	15 ／17	13 ／15
整数命令	1	1
（分離／結合）	1 ／1	1 ／1

単位：クロック

さらに平方根の演算では初期値テーブルを用意して高速化を図り，それぞれ 13,15 クロックで演算する．

表 2 に命令実行レイテンシとスループットをまとめる．FPU は 2M ゲートクラスの FPGA(XC3S2000:Xilinx) 1 個に実装，拡張 URR を扱う場合は，分離回路，結合回路を再構成する．

4．分離，結合ステージの隠蔽

演算パイプのレイテンシの不利は基本的に解消されるので，ロード命令に続く分離ステージ，演算命令に続く結合ステージの隠蔽が課題になる．

分離が隠蔽できないケースは，ロード命令が浮動小数点レジスタにロードするデータを後続の演算命令が直ちに使用する場合である．これは，通常は命令実行のスケジューリングで回避できるが，スケジューリングができない場合にのみ分離に起因する 1 クロックのストールが発生する．また，結合が隠蔽できないケースは，先行の演算命令が書き替える浮動小数点データを後続のストア命令が直ちにストアの対象とする場合で，これも通常は命令実行のスケジューリングで回避する．しかし，スケジューリングができない場合は，結合に起因する 1 クロックの余分のストールが発生する．もちろん，信頼性の高い評価を得るには，多くのベンチマークプログラムを使用して評価されなければならないが，後述の Graeffe 法による 4 次方程式の根の計算プログラムでは，そのほとんどが隠蔽されている．

5．評価

Graeffe 法は，繰り返しにより係数が急激に大きくなり，IEEE 標準の浮動小数点数では十分な繰り返し演算が困難になることが多い．ここでは 4 次方程式

$$(x-2)(x-e)(x-\sqrt{7.4})(x-3)=0$$

の根を計算するプログラムを作成して実装，スループットと，URR と拡張 URR との比較で，計算した根に含まれる相対誤差の評価を試みた．

（1）プログラム

4 次方程式の 5 個の係数と定数 1 個を浮動小数点レジスタに読み込み，浮動小数点レジスタ上で Graeffe 法による根の計算を行う．設定した回数の繰り返し演算を行い，4 個の根を絶対値で求めるところまでを実装した．

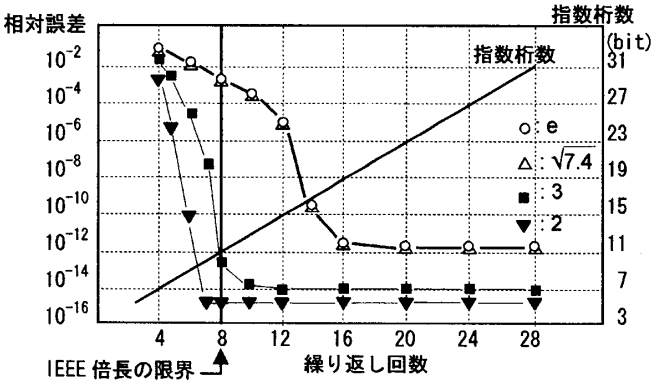

図 3．演算結果の相対誤差(Graeffe 法)

（2）スループット

プログラムが実行する各命令の数とその命令のスループットの積の総和から，プログラムの実行に要するクロック数を大まかに見積もった．繰り返しを 1 回で終了する場合，最初の命令フェッチを含めて，必要なクロック数 147 クロックが見積もられるのに対して，実機上でのカウントは 148 クロックであり，ほぼ一致する．増加した 1 クロックは，この場合は正規化ユニットの競合による命令フェッチ 1 ステージのストールであり，6 個の分離ステージ，31 個の結合ステージは，ほぼ隠蔽されている．

（3）相対誤差

4 次方程式の根の計算では，係数を浮動小数点レジスタにロードした後は，全ての計算を浮動小数点レジスタ上で行うことができる．浮動小数点レジスタには指数と仮数が分離されて格納されるので，指数の桁数の増加は仮数のビット数に影響しない．浮動小数点レジスタ上では URR，拡張 URR の区別はなく，図 3 に示すように，計算が収束した後は，指数のビット数が増加しても相対誤差はフラットである．

しかし，アプリケーションによっては，指数が大きい領域でロード・ストアが実行されることも考えられる．そこで，プログラムを改変して演算ごとにロード／ストア（指数が大きい領域でも分離結合が繰り返される）を行い，拡張 URR との比較を行った．その結果，URR では相対誤差が一度最小となった後，繰り返しにより漸増する傾向があるのに対し，拡張 URR では，図 3 の範囲ではわずかな増加に止まっており，改善されることが実機上でも確認された．

6．まとめ

URR，および拡張 URR を演算の対象とする 64 ビット FPU を，FPGA を用いて実現した．指数と仮数の分離・結合ステージを演算パイプから削除することにより，演算パイプのレイテンシを短縮した．さらに，演算パイプの外で行う分離と結合を，他のステージとオーバラップさせることにより可能な限り隠蔽，スループットの最大化を図った．実装した Graeffe 法による 4 次方程式の根の計算プログラムの事例でもほぼ隠蔽している．信頼性の高い評価を得るには，さらに完成度を高めた FPU を実現して，多くのベンチマークプログラムでの評価を積むことが欠かせないが，IEEE 標準の浮動小数点数を演算する FPU に迫るスループットを得る，という目標への到達の可能性の一端は示せたと考えたい．

【参考文献】

[1]浜田穂積：2 重指数分割に基づくデータ長独立実数値表現法 II，情報処理学会論文誌，Vol. 24, No. 2, pp. 149-156(1983)．

[2]大山光男：レイテンシを短縮した 32 ビット URR 浮動小数点数演算器の設計と実装，B-028, pp. 157-158, FIT2005(2005)．

[3]富松　剛：拡張した二重指数分割表現による数値表現法に関する研究，情処研報，95-HPC-58, pp. 57-62(1995)．