B-028

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

Design and Implementation of a 32bit Floating-Point Unit 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浮動小数点数（以下URR)は，事実上オーバーフローもアンダーフローも発生しないこと，フォーマットがデータ長独立であること，などの好ましい特徴を持つ[1]．しかし，IEEE標準など，固定長の指数部を持つ浮動小数点数の演算に比べて，指数と仮数の分離と結合に手間がかかることから，演算パイプにおいて各１ステージが余分に必要となる[2]．レイテンシの増加は，スループット低下の要因となり得るので，できるだけ短いことが望ましく，報告者らは，URR演算器のスループット改善を目指して，レイテンシを短縮した演算器の検討を行ってきた[3]．

本稿では，演算パイプから指数と仮数の分離と結合のステージを削除し，レイテンシを短縮したURR演算器の設計と実装結果について報告する．

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．

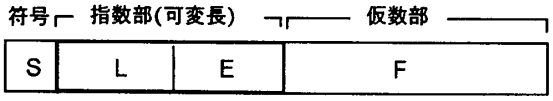

図１．　URR浮動小数点数のデータフォーマット

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

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

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

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．

3．レイテンシ短縮の方法

IEEE標準など，固定長指数部を持つ浮動小数点数の演算器のレイテンシに近づけるため，以下の方式を検討した．

（１）分離・結合ステージを演算パイプから削除

そのために，浮動小数点レジスタには，指数と仮数を分離して格納する．演算パイプは，分離・結合ステージを含まず，演算手続き上は指数部固定長の浮動小数点数の演算と同等のレイテンシが期待できる．さらに，データが演算器内に留まっている限り，指数の大きさに関係なく仮数の長さが維持されるという利点がある．

（２）指数と仮数の分離・結合のロード／ストア命令へのリンクと結合の投機的実行

演算パイプから削除した，指数と仮数の分離・結合回路をどこに置くかが問題となる．ここでは，浮動小数点レジスタとデータメモリ（キャッシュ）との間に置き，ロード命令の実行時に分離を行い，ストア命令の実行に先行して結合を行う．結合に関しては，演算結果で浮動小数点レジスタを更新する時，同時に結合を行い，浮動小数点レジスタとは別に設けたバッファレジスタに格納しておく．一種の投機的な実行になるが，ストア命令の結合ステージを省略でき，結合結果がバッファレジスタにあれば，待ちも発生しない．

４．演算器の設計と実装

以上の検討結果に基づいて32ビットの演算器を設計，FPGAを用いて実装し，その動作を確認した．表１に試作した演算器仕様の概略を示し，図２に内部構成を示す．

表１．演算器仕様の概略

浮動小数点数	URR 3重指数分割に基づく浮動小数点数
分離	32bit浮動小数点数 → 指数16bit 仮数32bit
結合	指数16bit → 32bit浮動小数点数 仮数32bit
命令セット	浮動小数点命令：加算，減算，乗算， 転送，ロード，ストア，他
浮動小数点 レジスタ	(16+32)bit×16本

（１）演算器の仕様

URRに加えて３重指数分割に基づく浮動小数点数をあつかう[4]．分離・結合回路の変更が必要であるが，これはFPGAでは容易に再構成できる．32ビットの浮動小数点数からは16ビットの指数と32ビットの仮数を分離する．結合では，この逆の処理を行う．命令セットは，ごく基本的な演算を行う浮動小数点命令に加えて，簡単なプログラムが実行できるよう，整数演算命令を備えている．

† 倉敷芸術科学大学　コンピュータ情報学科

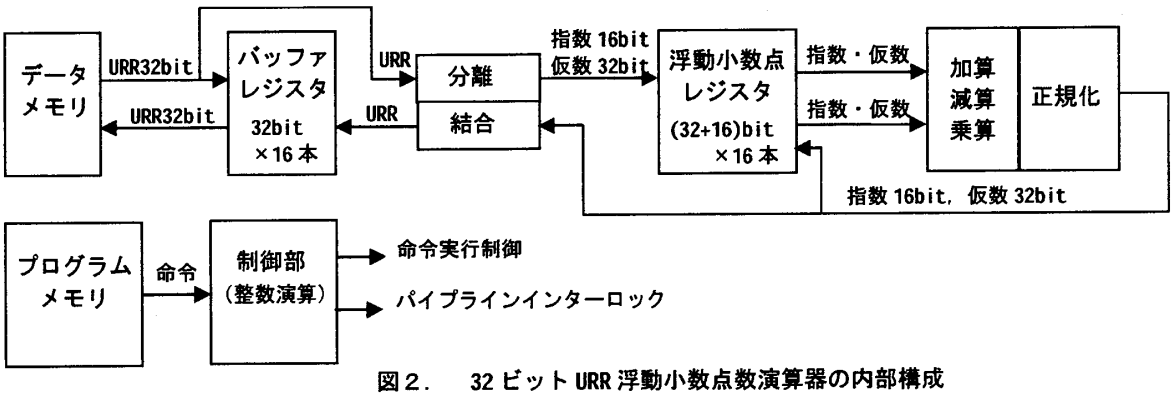

図２．　32 ビット URR 浮動小数点数演算器の内部構成

（２）内部構成

図２に示すように，浮動小数点レジスタに指数と仮数を分離して格納するので，演算パイプは分離・結合回路を含まない．よって演算パイプには，指数部固定長の浮動小数点数の演算に比べて余分な遅延要素は基本的に存在しない．

分離回路と結合回路は，浮動小数点レジスタとデータメモリの間に置く．バッファレジスタの各レジスタは，浮動小数点レジスタの各レジスタに対応しており，同じデータが URR として格納される．

（３）動作

浮動小数点演算命令では，浮動小数点レジスタから指数と仮数が同時に読み出され，演算，正規化後，結果の指数と仮数が浮動小数点レジスタに書き戻される．演算結果は同時に，結合回路にも入力され，引き続く１クロック（１ステージ）で URR に結合，対応するバッファレジスタに格納する．

ロード命令では，データメモリから URR を読み出し，バッファレジスタに格納すると同時に，同じデータを分離回路にセット，引き続く１クロックで指数 16 ビット，仮数 32 ビットを分離して浮動小数点レジスタに格納する．

ストア命令では，データメモリに書き込むデータがバッファレジスタにあれば，直ちに読み出してデータメモリに書き込み，無い場合は URR の生成を待つ．なお、図２には表していないが，データメモリへの書き込みデータとして，結合回路の出力を選択するパスを設ければパイプのストールが避けられる場合がある．

（４）実装

図２に示す演算器を Xilinx 社の FPGA，XC3S400 を用いて実装，30MHz のクロックでの動作を確認した．表２に実装結果を示す．

表２．実装結果

	URR	3 重指数分割に基づく浮動小数点数
ハードウェア規模（4 入力ルックアップテーブル数）	Total: 2192 分離:　437 結合:　339	Total: 2140 分離:　344 結合:　378

註：（１）デバイス：XC3S400 (Xilinx)
（２）乗算器，メモリ，浮動小数点レジスタ，バッファレジスタには，埋め込み乗算器，ブロックメモリを使用．

６．スループットの見積もり

ベンチマークプログラムによる評価が望ましいが，充分な命令セットが実装されていないので，今回は，図２の構成をベースに，分離・結合を演算パイプに含む場合，および IEEE 標準（分離・結合とバッファレジスタ無し）との比較を行った．ロード，ストア，浮動小数点演算，整数演算の各命令実行のレイテンシは，それぞれ 2,1(+1),2,1（本演算器），1,1,4,1（演算パイプに分離・結合を含む），1,1,2,1(IEEE)とした．データメモリ上の配列に対する積和演算ループなど，限られた事例の見積もりではあるが，演算パイプに分離・結合を含む場合に比べて 20～30%改善，IEEE 標準に迫るスループットを見込んでいる．

５．まとめ

指数と仮数の分離・結合を演算パイプから削除することにより URR の演算パイプのレイテンシは，IEEE 標準など，固定長の指数部を持つ浮動小数点数の演算に比べて，演算手続き上は同等となる．さらに，演算パイプは基本的にデータフォーマットに独立となり，分離・結合回路の再構成により，複数の可変長指数部をもつ浮動小数点数の演算ができる．また，演算器にデータが留まる限り，仮数の長さが指数のサイズの影響を受けることもない．一方，削除した分離・結合機能はデータメモリ（キャッシュ）と浮動小数点レジスタ の間に置くことになり，新たなペナルティとなり得るが，結合の投機的実行など，分離・結合を先行して行うことにより影響の軽減が期待できる．

現在，命令セットを強化した 64 ビット FPU を開発中であり，より実用的なプログラム，環境での評価を行いたい．

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:URR 浮動小数点数演算のためのパイプライン加減算器の設計と FPGA による実現，情報処理学会研究報告，97-HPC-69,pp19-24(1997).

[3]Mitsuo Ooyama:" A Floating-Point Unit with Reduced Latency for URR Floating-Point Arithmetic" ,in SCAN2004,p.92(2004).

[4]中森真理雄，土井　孝：３重指数分割による数値表現方式について，電子情報通信学会論文誌，Vol.J71-A,No.7,pp1468-1469(1988).