ハイパフォーマンス
コンピューティング 69－4
（１９９７．１２．１２）

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

大山 光男

倉敷芸術科学大学　ソフトウェア学科
〒712　岡山県 倉敷市 連島町 西之浦 2640番地
(e-mail : ooyama@soft.kusa.ac.jp)

URR浮動小数点数と３重指数分割に基づく浮動小数点数の高速演算を目的として，４ステージパイプライン加減算器を設計した。指数と仮数の分離，仮数の桁合わせと加減算の実行，演算結果の正規化，指数と仮数の結合を，それぞれ１ステージで実行する。さらにパイプライン加減算器の機能を確認するためFPGAに実装して評価した結果，クロック周波数12MHzでURR浮動小数点数の演算を，10MHzで３重指数分割に基づく浮動小数点数の演算を確認した。

Design of a Pipelined Adder/Subtracter for URR Floating-Point Arithmetic and its Experimental Implemention on FPGAs.

Mitsuo OOYAMA

Department of Computer Science and Mathematics
Kurashiki University of Science and the Arts
2640 Nishinoura, Tsurajima, Kurashiki, Okayama 712 JAPAN
(e-mail : ooyama@soft.kusa.ac.jp)

Four-stage pipelined adder/subtracter for URR floating-point arithmetic and floating-point arithmetic based on a triple exponential cut was designed. Separating an exponent and a fraction from URR, alignment of a fraction and addition/subtraction, normalization of the result, and combining an exponent and a fraction into URR are processed on each stage of the pipeline. This adder/subtracter was also implemented on FPGAs and correct calculation of URR with 12MHz clock and floating-point arithmetic based on a triple exponential cut with 10 MHz clock were confirmed.

1　はじめに

本稿では，可変長指数部を持つ浮動小数点表現であるURR，およびその変形である３重指数分割に基づく浮動小数点数を演算するパイプライン加減算器の設計と，FPGAへの実装結果について報告する。浜田によって提案されたURRは事実上オーバフローやアンダーフローを起こさないほか，いくつかの望ましい特徴を持っている(1)。しかし，指数の絶対値が大きい（桁数が多い）領域では指数部が長くなることから，その改善がいくつか試みられている(2)(3)(4)。

一方，これらの数値表現の実用化が促進される条件の一つに，高速演算機構が安価に提供されることがある。URRは，他の表現範囲の広い実数値表現，例えば，指数関数の計算が必要となるLI (Level Index)(5)等に比べて簡単な操作で演算できるが，指数部固定長の浮動小数点数の演算比較して指数仮数の分離結合に手間がかかる，などの課題がある。筆者らは，高速演算機構の実現を目的として，そのキー技術になると思われる指数と仮数の高速分離結合回路方式の検討(7)(8)と，マイクロプログラム制御方式のURR演算プロセッサへの応用を行ってきた(6)(7)。しかしながら，本格的な高速演算機構を実現するには，高速なパイプライン演算器の開発が必須となる。本稿ではその１ステップとして設計試作した，４ステージで32ビットURR，および３重指数分割に基づく浮動小数点数を演算することのできるパイプライン加減算器について報告する。

2　表現の概要

最初にURR，および3重指数分割に基づく浮動小数点表現のデータフォーマットについて概要を述べる。なお，以降では3重指数分割に基づく浮動小数点表現を，3重指数分割，あるいはTriple exp. cutと省略して記す。

（1）URRのデータフォーマット

図1に示すように，符号ビットに続く指数部は，指数の値を保持するE部と，E部の長さを表すL部から成る。したがって指数部は可変長であり，固定長データでは残りが仮数部となる。各部を構成するビットは以下のように決められる。

いま，実数xを

$$x = 2^e \times f$$

指数eは整数，仮数fは

$$1 \leqq f < 2 \quad (x \geqq 0)$$

$$-2 \leqq f < -1 \quad (x < 0)$$

と表す。図1において，符号ビットSは仮数fの符号であり，仮数部は仮数fの小数点以下のビット列が上位から順番に入る。指数部は，指数eがm桁で

$$e = \cdots S_e \overline{S_e}\ e_{m-1} \cdots e_1 \quad , S_e\text{は符号}$$

と表され，x≧0とき，

$$\overbrace{1\cdots10}^{L}\ \overbrace{e_{m-1}\cdots e_1}^{E} \quad (e \geqq 0)$$

（m+1個）

$$\overbrace{0\cdots01}^{L}\ \overbrace{e_{m-1}\cdots e_1}^{E} \quad (e < 0)$$

（m+1個）

となる。なお，e＝-2,-1, 0,1の各場合は，E部は空であり，L部はそれぞれ001, 01, 10, 110である。

x<0の場合は，上記ビット列の1/0を反転したビット列（1の補数）が指数部となる。

（2）3重指数分割のデータフォーマット

図2において，符号Sと仮数部はURRに同じであるが，指数部がL,N,E部から成る。EはURRに同じであるが，N部が数値でE部の長さ（指数の桁数）を表し，L部がビット列の長さでN部の長さを表す。すなわち，

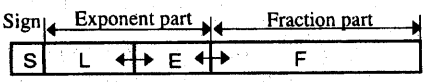


Fig. 1　Data format of URR

図1　URRのデータフォーマット

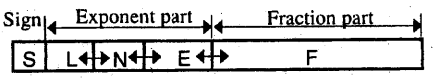


Fig. 2　Data format of FLP based on Triple exp. cut.

図2　3重指数分割に基づく浮動小数点数のデータフォーマット

指数eがm桁で表され，x≧0のとき，指数部は

$$\overbrace{1\cdots10}^{L}\ \overbrace{n_{k-1}\cdots n_1}^{N}\ \overbrace{e_{m-1}\cdots e_1}^{E} \quad (e \geqq 0)$$

（k+1個）

となる。e<0では上記指数部のL,N部の1の補数をとり，x<0ではさらに指数部全体の1の補数をとる。またe＝-2,-1,0,1の各場合は，N部，E部は空であり，指数部はURRと同じである。

3　指数と仮数の分離

次に指数と仮数の高速分離結合回路方式について要点を述べる。ここでは，URR，3重指数分割とも，分離においては分離前と同一長の指数と仮数を分離し，結合では同一長の指数，仮数から同一長のURR，3重指数分割を得ることとする。3重指数分割では指数の長さを制限することになるが，これにより32ビットデータのN部の長さは高々4ビット，64ビットデータでも5ビットと極めて短くなり，N部の分離が高速，かつ少量のハードウェアで可能となる。

3.1　URRからの指数，仮数の分離

次の手順により行う

（1）各部の位置と長さの検出

具体的にはL部の右端ビット（区切りビットの位置を検出し，左端の2ビットに検出したビットのビット番号（LSBが0）を連接し，境界コード(B-code)を生成する。区切りビットが存在しない場合は，ビット番号を最大（左端ビットの番号）とする。L部は必ず存在し，L部の長さが分かれば，E部，F部の有無と長さが分かる。高速検出は，プライオリティエンコーダを利用したハードウェアにより可能である。

（2）指数の分離

E部に先行するビット列を指数の符号ビットに変換後，指数のLSBが右端に位置するよう算術右シフト（仮数部がないときは左）する。指数の符号ビットへの変換は，境界コードから生成するビットパターン1とURRとのXOR演算で可能であり，シフト量も境界コードから決まる。なお，32ビットURRに対するビットパターン1の詳細を，後掲する表2に示した。

（3）仮数の分離

仮数部の先頭ビットが左端から4ビット目に位置するように左シフトし，左端3ビットを，2ビットの符号ビットSとその反転ビット1ビットで置換する。シフト量は境界コードから決まる。

以上の手続きは図3に示す回路で高速に実行できる。図に示すように境界位置検出後は，指数の分離と仮数の分離は並列に実行する。

3.2　3重指数分割からの指数と仮数の分離

最初にN部を分離する。これはURRにおける指数の分離と同様な手順で行うが，N部は短く抑えられているので容易である。次に分離したN部と境界コードを用いて指数と仮数を分離するが，以降の手続きはURRと同様である。

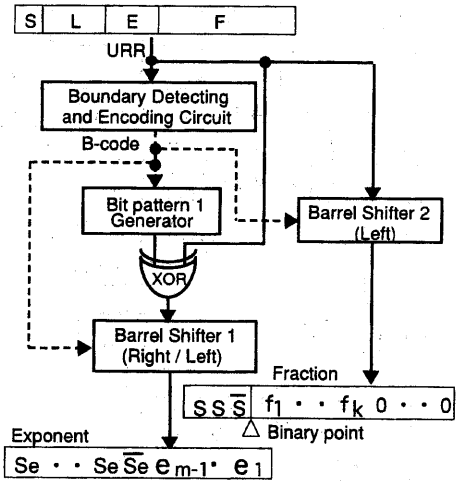


Fig.3 Block diagram of the separating circuits (For URR)

図３　指数と仮数の分離回路の構成（URR)

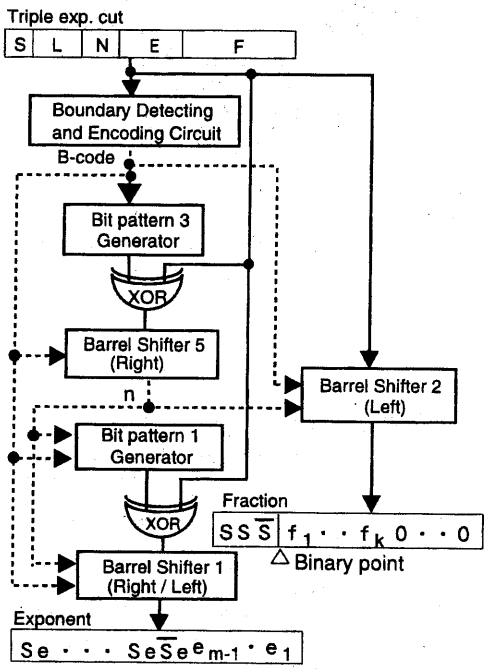


Figure 4　Block diagram of separating circuits (For Triple exp. cut).

図４　指数と仮数の分離回路の構成（3重指数分割）

なお，32ビットの３重指数分割から指数を分離するためのビットパターン1の詳細を後掲の表３に示した。

以上の手続きは図４に示す回路により高速に実行される。

4　指数と仮数の結合

指数と仮数に分離，演算を行った後は，指数と仮数を結合してURRや３重指数分割を生成する。以下ににその手順を述べる。

4.1　指数と仮数の結合（URR)

（１）指数の桁数の検出

指数の左端の符号ビットを基準に最初の反転ビットの位置を検出し，そのビット番号を指数コード（E-code）として出力する。ビット反転がない場合は最大ビット番号をあてる。指数の桁数が分かれば生成するURRの各部の位置，長さが全て確定する。

（２）指数部の生成

指数の桁数で決まる量，指数を左シフトして，URRのE部に合わせる。このとき，仮数部が存在しない場合は右シフトとなる。次に指数コードからビットパターン2を生成して，シフトした指数とXOR演算を行いL部，E部を生成する。このとき，左端ビットは空にする。ビットパターンの詳細は後掲の表４に示した。

（３）仮数部の生成

指数の桁数で決まる量，仮数を右シフトし，

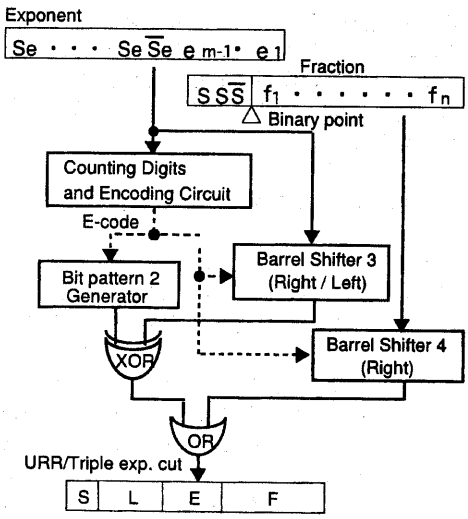


Fig. 5　Block biagram of the combining circuits (For URR and Triple exp. cut.)

図５　指数と仮数の結合回路の構成

URRの仮数部にあわせる。このとき，仮数部に先行する指数部となる位置は空（ゼロ）とする。

（4）（2），（3）で生成した指数部仮数部をOR演算で結合してURRを生成する。

以上の手続きは図5に示す回路で高速に実行する。高速化のため指数の桁数検出後は指数部と仮数部の生成を並列に行う。

4.2 指数と仮数の結合（3重指数分割）

生成する指数部の構成は異なるが，URRと同一手続きで結合出来る。3重指数分割の指数部を生成するにはビットパターン2を入れ替えればよい。32ビットの場合のビットパターン2を後掲の表5に示した。結合回路はビットパターン2を入れ替えて図5の回路が利用できる。

5. パイプライン加減算器の設計

前述の分離回路，結合回路を利用してパイプライン加減算器を構成する。指数と仮数の分離後は，指数，仮数のそれぞれで演算を行い，結果を再び結合する。指数と仮数の演算は，指数の長さが長いことを除けば従来の浮動小数点数演算器と同様であり，従来の回路が使用でる。従って，パイプライン加減算器は図6に示すように構成される。パイプラインの各ステージへの処理の割り当てでは，ゲートアレイの設計パラメータで遅延時間を見積もり，出来るだけバランスが取れるようにした。その結果，図7に示すように，指数と仮数の分離，仮数の桁合わせと加減算の実行，演算後の正規化，指数と仮数の結合を各1ステージで行い，4ステージのパイプライン処理が可能なことが分かった。なお，丸めに関しては，ここでは切り捨て処理とし，行わない。

6 FPGAによる実現

図6に示すパイプライン加減算器の機能を確認するため，FPGAを用いて試作，評価を行った。ゲート数と入出力ピンの数から必要となるFPGAの個数を見積もり，データ長は32ビットとし，URRと3重指数分割の演算の切り替えは，FPGAのプログラムを替えることにより（具体的にはROMの交換）実現とすることにした。使用したFPGAはXILINX社のXC4013E-3，利用したデザインCADは，IBM互換のパソコン上で稼働する同社のXACT-STEP V6.01である。

実装結果を示す前に，ハードウェア量の多い回路の構成を次に説明する。

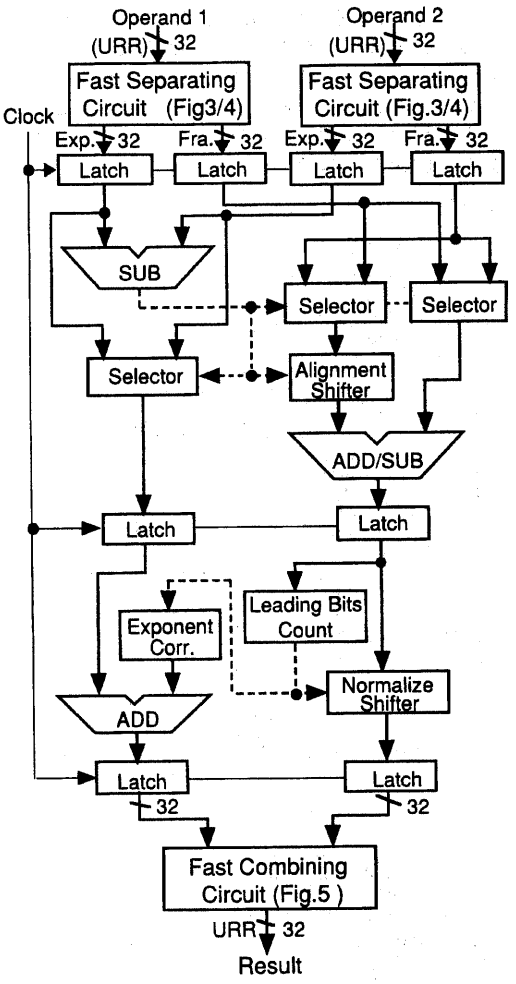
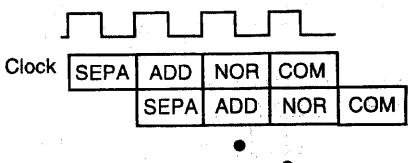


Fig. 6 Circuit diagram of a pipelined adder/subtracter for URR floating-point arithmetic.

図6 パイプライン加減算器のブロック図



Clock: 12MHz (URR)
10MHz (Triple)
SEPA: Separation
ADD: Alignment and Addition/Subtraction
NOR: Normalization and correction to an exponent
COM: Combination

Fig. 7 Pipeline stages

図7 パイプラインステージ

多数のバレルシフタが存在するが，ハードウェアの増加を極力抑えるため，3段，または4段の階層構成とした。ビットパターンは論理回路で生成した。これは今回，FPGAを使用したことにもよるが，表2～5に示すように，ビットパターンが比較的規則的であり，やや変則的な3重指数分割においても多量の論理回路は必要としないからである。ただ他の方式との定量的比較評価は行っておくべきかもしれない。加算器と減算器の構成ではFPGAが備える専用の高速キャリー伝搬回路を利用した。

図8に示す全回路を5個のFPGAに分割して実装した。分離回路，仮数の演算回路，指数の演算回路，結合回路をそれぞれ1個のFPGAに実装した。1個のFPGA(XC4013E)は論理機能の実装単位であるCLB(Configurable Logic Block)を576個備えており，各CLBは図8に示すように2個の4入力，1個の3入力論理ファンクションジェネレータと2個のフリップフロップなどから構成されている(9)。各FPGAごとにファンクションジェネレータ，CLBの使用個数（使用率）を表1に示す。なお，パイプラインステージを構成するためのフリップフロップを除いて，回路は組み合わせロジックで構成されるため，CLBのフリップフロップはほとんど利用していない。

以上の結果，クロック周波数12MHzでURRの加減算が，10MHzで3重指数分割の加減算の実行がが確認された。

7 考察

（1）ハードウェア量に関して

従来の指数部固定長浮動小数点数の演算器に比べて，URRや3重指数分割の演算ハードウェアが増加する主たる要因は，指数と仮数の分離結合に要するハードウェアと，長くなりうる指数を処理するためのハードウェアと考えられる。今回のパイプライン加減算器では，仮数と指数の演算回路が最低限の機能しか実装していないことや，専用の高速キャリー伝搬回路の利用によりハードウェアが少な目であることを考慮しても，指数仮数の分離結合回路のハードウェアはかなり大きいと考ねばならない。FPGAの実装の内訳はまだ得られていないが，ゲートアレイでの見積もりではバレルシフタのハードウェア量が約2／3を占めるとのデータが得られている。須田らの指摘(10)にあるようにハードウェアを大幅に削減するにはシフタの問題を解決する必要があるかも知れない。

（2）パフォーマンス

今のところ指数仮数の分離，結合にかかる時間を無視できるレベルにするには無理があるが，分離と結合に各1ステージをあてたパイプライン処理ができる可能性があると考えられる。FPGAでは配線のプログラム機能に起因する遅延時間が大きいので，パフォーマンスの限界はセルベースLSIなどで見積もる必要があろう。

8 おわりに

指数と仮数の分離と結合に各1ステージをあてた，4ステージ，32ビットのURRと3重指数分割に基づく浮動小数点数を演算するパイプライン加減算器を設計した。また，この演算器をFPGAに実装した結果，クロック周波数12MHzでURRの演算を，10MHzで3重指数分割に基づく浮動小数点数の演算を確認した。

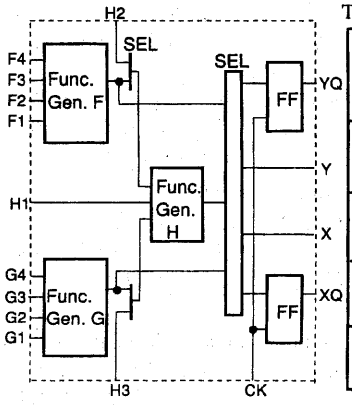


Fig. 8 Simplified CLB (XC4000E)

図8　CLB内部構成の概略

表1 FPGAへの回路の実装と各FPGAのファンクションジェネレータ／CLBの使用個数（使用率）

Table 1 Implemented circuits on FPGAs and used logic function generators/CLBs.

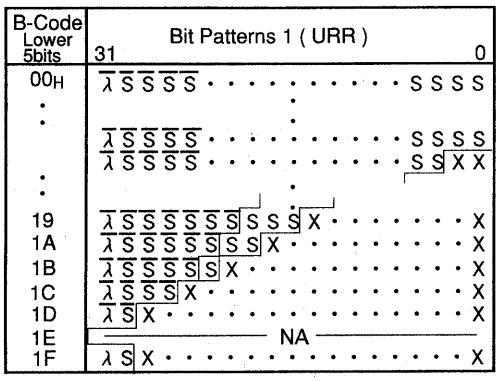
FPGA	Implemented Functions	Used Function Generators and CLBs					
		URR			Triple exp. cut		
		F/G	H	CLB	F/G	H	CLB
FPGA#1 FPGA#2	Separating Circuits (1 set each)	474 (41%)	161 (27%)	345 (60%)	619 (53%)	190 (32%)	427 (74%)
FPGA#3	Alignment Shifter Adder/Subtracter Normalizing Circuits	529 (45%)	144 (25%)	372 (65%)	←		
FPGA#4	Comparing Exponents Correcting Exponent	176 (15%)	19 (3%)	138 (24%)	←		
FPGA#5	Combining Circuits	500 (43%)	187 (32%)	340 (59%)	607 (52%)	237 (41%)	397 (69%)

FPGA: XC4013E-3 (Xilinx)

CLB:Configurable Logic Block, F/G: Logic Function Generator F/G, H: Logic Function Generator H

表2　URRから指数を分離するビットパターン1

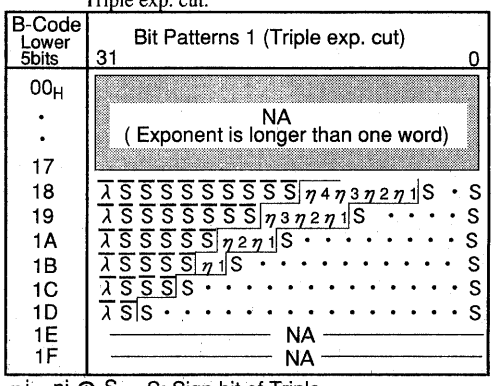
Table 2 Bit patterns 1 for separating an exponent from URR.



S: Sign bit of URR, λ : The first bit of the exponent part
X: Don't care bit

表3　3重指数分割から指数を分離するビットパターン1

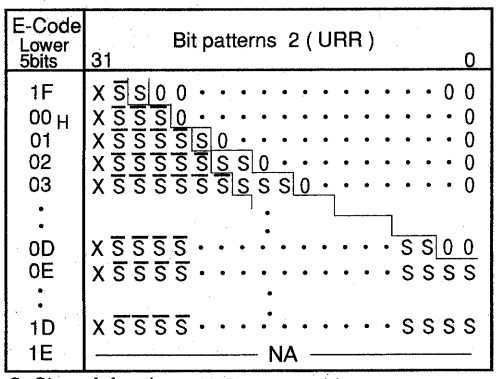
Table 3 Bit patterns 1 for separating an exponent from Triple exp. cut.



ηi = ni ⊕ S,　S: Sign bit of Triple
λ : The first bit of the exponent part

表4　URRの指数部を生成するビットパターン2

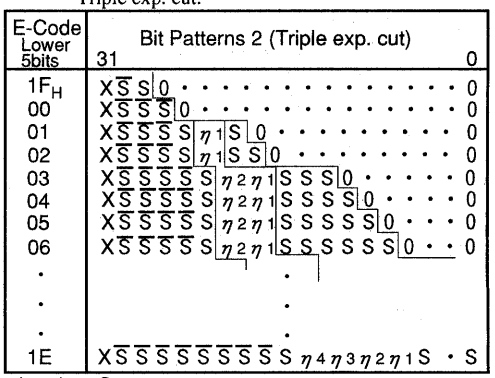
Table 4 Bit patterns 2 for generating an exponent part of URR.



S: Sign of fraction,　X: Don't care bit

表5　3重指数分割の指数部を生成するビットパターン2

Table 5 Bit patterns 2 for generating an exponent part of Triple exp. cut.



ηi = ni ⊕ S,　S: Sign bit of fraction
X: Don't care bit

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）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, pp.1468-1469 (1988).

3）横尾英俊：自然数の表現に基づく多重指数分割浮動小数点表示方式のクラス，電子情報通信学会論文誌，Vol.J72-A, No.12, PP.19982004 (1989).

4）富松　剛：拡張した二重指数分割による数値表現法に関する研究，情報処理学会研究報告，95-HPC-58, pp.57-62 (1995).

5）Olver F.W.J.,et al.:"Implementation of Level-Index Arithmetic Using Partial Table Look-Up"; Proceedings of the 8th. Symposium on Computer Arithmetic, PP.144-147 (1987).

6）中原雅彦，早川栄一，岡野裕之，並木美太郎，高橋延匡：複数の浮動小数点表現法を処理するシステム環境の設計と実現，情報処理学会論文誌，Vol.33, No.4, pp.481-490 (1992).

7）大山光男，浜田穂積：URR浮動小数点数演算のための指数仮数高速分離結合回路方式とそのURRプロセッサへの応用，情報処理学会論文誌，Vol.35, No.8, pp. 1642-1651 (1994).

8）大山光男：3重指数分割に基づく浮動小数点演算のための指数と仮数の高速分離結合回路の設計と評価，情報処理学会論文誌，Vol.37, No.4, pp.613-623. (1996).

9）XC4000E Field Programmable Gate Array Family Preliminary Product Specifications, Xilinx, Sept. 1995.

10）須田礼仁，小柳義夫：新しい可変長指数部表現形式の提案，情報処理学会研究報告，97-HPC-66, pp.31-36 (1997).