

FPGA 向け逆数計算回路の設計

尾形 航 笠原 博徳

早稲田大学 理工学部 電気電子情報工学科
〒169-8555 東京都 新宿区 大久保 3-4-1
TEL 03-3209-6323
FAX 03-3232-3594
e-mail: {ogata,kasahara}@oscar.elec.waseda.ac.jp

あ ら ま し 計算機アーキテクチャ研究において対象アーキテクチャを評価するには、ソフトウェアでエミュレートを行う、あるいは実際にハードウェアを製作して評価する等の手法が取られてきた。しかし前者の方法では評価に膨大な時間を要するという難点があり、一方後者の実際に製作する方法では多大な費用がかかる、開発から実際に稼動して評価に入るまでに相当の期間を要する、また一度製作した機体を改造して別のアーキテクチャの評価を行うことが事実上不可能、等の問題があった。これらの問題を解決する高速・浮動小数点演算機能を含む高性能、過大でない費用で実現可能、容易にアーキテクチャを改変可能なハードウェアアーキテクチャエミュレータを、S-RAM 型の大規模な FPGA を用いて開発した。この上で実用に即したアプリケーションを走行させる際には浮動小数点の除算も多用されるが、これを効率よく実行する為に、FPGA の回路の特徴を活かした逆数計算回路を設計する。

キーワード 浮動小数点演算 FPGA エミュレーション 評価 ベンチマーク 計算機アーキテクチャ

A Reciprocal Number Calculation Circuit Design for FPGA

W.Ogata H.Kasahara

Department of Electrical, Electronics and Computer Engineering, Waseda University
55N-05-05A, 3-4-1 Okubo, Shinjuku-ku Tokyo 169-8555 Japan
TEL: +81-3-3209-6323
FAX: +81-3-3232-3594
E-mail: {ogata,kasahara}@oscar.elec.waseda.ac.jp

Abstract

There are two major approaches to evaluate new computer architecture, the evaluation with software emulator on WS or high-performance PC. The other is the evaluation on hardware emulator. However, it takes very long time on the evaluation with the software emulator to evaluate new architecture using large benchmark programs. On the other hand, it is expensive to develop a machine. Also, it takes long time to develop new machine. Even if we have hardware, it is very difficult to modify it. So, evaluation of different architecture is difficult by using the same hardware. To cope with these problems, we have been developed an architecture emulator, which is high-performance with floating point arithmetic unit; not so expensive cost, reconfigurable easily. It is combination of S-RAM based large scale FPGAs(Field Programmable Gate Array). During the execution of real-benchmark program, many amount of floating division operation may be executed. So we have designed a reciprocal number calculation circuit considering the feature of FPGAs.

key words

Floating-Point Calculation FPGA Emulation Evaluation Benchmark Computer-Architecture

1. はじめに

計算機アーキテクチャを新たに提案したり、新たなアーキテクチャと組み合わせるコンパイル手法を提案する際には、実用に即したアプリケーションを集めた SPEC や PerfectClub 等のベンチマークプログラムを用いて評価を行なう必要がある。しかしソフトウェアエミュレーションでは速度が遅く Pentium-II(233MHz)上で単純なエミュレータを走行させた場合には 200KHz 程度、メンテナンス性を重視して C++で記述したエミュレータでは 10KHz 程度の性能となり、数万行に及ぶ本格的なアプリケーションで一つのケースについて評価を行うのに数日~数週間を要する事も珍しくなく研究に障る。一方実際に対象アーキテクチャに沿ったマシンを製作するには多大な費用を要する他、設計を開始して製作・調整を経て実際に評価に入れるようになるまでに相当の期間を要する難点がある。そして一度製作した機体の設計を変更して他のアーキテクチャを試行することは非常に困難であり、アーキテクチャ研究の上では好ましくない。

これらの難点を解消するために

- ・高速である
- ・浮動小数点演算機能など高機能を持つ
- ・容易に改変可能である
- ・改変後素早く評価にうつれる
- ・機能維持・保守が容易である
- ・過大でない予算で実現可能である

特徴を持つハードウェアアーキテクチャエミュレータを製作した。ここでは S-RAM 型大規模 PLD の LUT の特性を活かして浮動小数点演算回路をコンパクトに実現した本格的な 32bitRISC 型 CPU の実装している。

このエミュレータ上で実用に即したベンチマークテストを走行させる場合には、浮動小数点演算も多用され、その中でも除算の頻度は決して少なくない[1]。除算を行うには、プログラムで 1 ビットずつ商を得る方法、マイクロプログラミングに依る方法、ワイヤードロジックとして組む方法などがある。中でもマイクロプログラミングに依る方法では、1 ビットずつ商を得る方法や、SRT 法[2]による方法などがあるが、ステップ数が多く、速度を向上させることは困難である。またワイヤードロジックを用いる場合にも効率の良い回路の例はなかなか見られない。

本稿では、FPGA の LUT 並びにメモリ機能を用いて、小規模な表と、小規模な演算回路を組みあわせて逆数計算を行う回路の設計について述べる。その一つは N/R 法[1]をベースとして 1K ビット程度の容量のメモリに格納した表を基に初期値を計算して、更に反復計算を行って単精度で逆数を求める方法、いま一つは、より大きな表を 4~6 個用いて直接単精度の逆数を求める方法である。

2. ハードウェアアーキテクチャエミュレータ

既に実装した、大規模 PLD をベースとしたハードウェアアーキテクチャエミュレータについて下記に説明する。

2.1. 使用デバイスの選定・実装

本研究においては単に対象となるアーキテクチャを実現するのみならず、演算回路の比較検討などを試行する特性が求められる。市場にはビュース型乃至アンチビュース型で廉価且つ高速の機種があるが再書き込みが不可能で本研究には適さない。或は紫外線消去型やフラッシュ型で高速の機種もあるが書換え回数に制限があったり専用の書き込み機器へ差替えるなど取り回しの面で難がある。本研究では書換え回数に制限が無く、書込時に専用の機器へ差替える必要の無い S-RAM 型の機種が対象となる。ここでは ALTERA 社の FLEX10K シリーズの内、設計開発時に最大であった 10K100(4 入力 LUT4992 個、公称値 10 万ゲート相当)を使用した。

この FPGA を実装する試験用モジュールを写真に示す。EPF10K100 に、S-RAM、SIMM(D-RAM)を搭載している(写真1、写真化学製 AM4-RAM 型)。

写真1

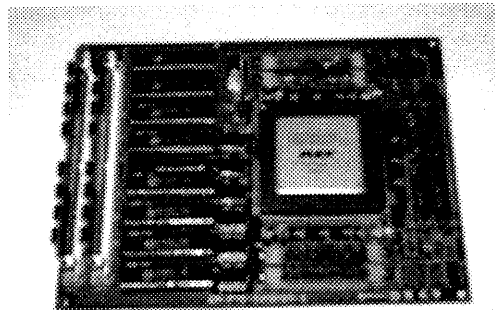
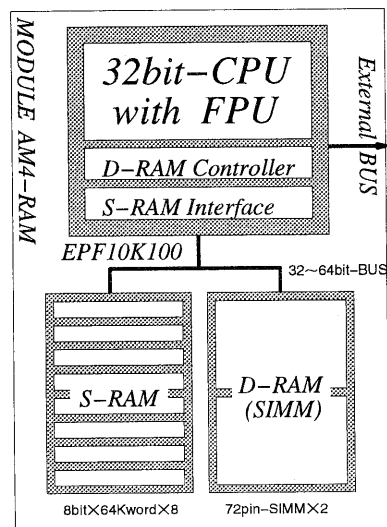


図1



2.2 SPARCの実装

このFPGA上に、浮動小数点演算機能付きのSPARCのサブセット(倍精度演算等を除き、レジスタ間演算、ロード/ストア、分岐等に限る)を実装する。SPARCを選択したのは、汎用として数が出回っておりアーキテクチャの比較対象として適当なこと、また、比較的単純な構造で、今後アーキテクチャの改良を施す際のベースとして適当なことがあげられる。

2.3 乗算回路の実装

このCPUに含まれる単精度の浮動小数点演算機能を含む演算回路をPLD上に実現するには、PLDの構成要素であるLUTに適した実装方法を考慮しなければならない。浮動小数点演算並びに整数演算における乗算は、演算回路の中で最も回路資源を要する。通常の加算を用いて32bit×32bitの演算を行なう場合には第一段で個々の部分積(32×32→1024項)を表現する為に1024個のLUTを使用する。そしてこれらの部分積を加え合わせる為に、第二段以後、各段に512, 256, 128, 64, 32個のLUTを使用し、全体で2000個強のLUTを使用する。この回路規模は他の回路を圧迫するので、よりコンパクトに乗算を実現するのが望ましい。

フルカスタムのLSI上に乗算回路を高速且つコンパクトに実装する際にBOOTHのアルゴリズム[3]ならびにWallaceツリー[4]が用いられる。例えば2次のBOOTHのアルゴリズムではアレイを配列する回路構成をとり、一つのアレイには×2, ×1の信号並びに符合Sの信号を組み合わせて×(+2), ×(+1), 0, ×(-1), ×(-2)を表現する。これを処理するには一つのアレイを実装する処理単位に5入力が必要である。ところが、使用するFPGAのLUTにおいては4入力を用いるので効率良く実装する事ができず、実装効率が却って低下する。

本稿ではLUTに適したアルゴリズムを用いてコンパクトに実現する。即ち単一のアレイで、当該ビットの×2, ×1, ×(-1)そして零を表現し、乗数の選択の為に4入力、×(+2), ×(+1)の値の入力に2入力、計4入力があればよい。×(-1)については、×(+1)の入力より2の補数を得て実現する。即ち各アレイにおいて×(+1)の1の補数を取り、後続段で部分積を加え併せる時に最下位ビットに1を加える、具体的には加算器の桁上がり入力に1を入力することにより全体で2の補数を実現する(図2)。これにより、乗算に要するLUTの個数を約半分に減らす事が出来る。

3. 除算(逆数計算)

浮動小数点除算は、演算の中でも最も時間を要する項目である。最も単純にはプログラムでシフト命令と減算/加算命令、条件分岐を組み合わせて1bitずつ商を算出する方法があげられるが、ループを処理する為に処理効率が著しく悪化する。特に1ステップを処理する度に条件分岐を処理する必要があるので、近年普及している命令パイプライン構成を持つプロセッサでは処理効率を向上

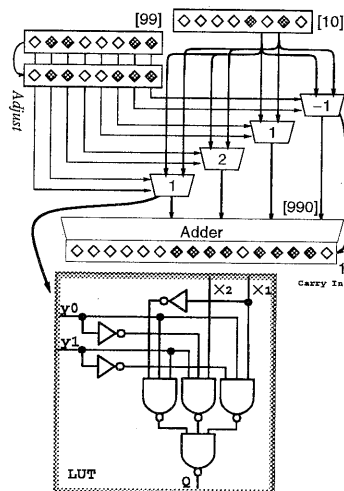


図2 4入力LUT上の乗算

させるのが非常に困難である。

より高速に除算を処理する手法としてSRT法がある。これは除数と被除数を上位から数ビットずつ取り、それを元に表を検索して数ビットずつ商を得る操作を繰り返して必要な精度の商を得る手法で、上記の1ビットずつ商を得る手法に比べて処理のステップ数を削減することが出来る。然し依然多くのステップ数を必要とする他、表を格納するメモリが必要となる点が難である。

また、上記の除算の操作をマイクロプログラミングによって処理する手法もあるが、この場合でも多数の処理ステップを要する事が多い他、別途マイクロコードをFPGA上に実装する必要がある、設計が複雑になる難点がある。

本稿ではニュートン・ラフソン法(以下N/R法と記す)を用いた、PLDに適した方式を実装する。

3.1 N/R法初期値の獲得

N/R法は与えられた除数の逆数の近似値 Y_0 を得て、これを種に反復計算を行なって所定の精度の逆数を得て、最後に逆数を被除数に乗じて結果を得る。N/R法において一回の反復で24ビットの精度を得るには、初期値 Y_0 として13~14ビットの精度が必要である。これを直接表として作成する場合には14ビット幅16384ワード、即ち229376ビット(230Kビット弱)容量のメモリが必要である。これは、フルカスタムLSIやゲートアレイ上に構成するには大きな面積を占有して非常に効率が悪い。また現行のFPGAの中で最大規模の機種を用いても、これだけの容量のメモリを搭載する事が出来ない。

3.2 表を用いたN/R法初期値の獲得

そこで、除数の定義域($0.5 \leq X_0 < 1.0$)を32区分して、各区分を1次近似する事で13~14ビットの精度で初期値を獲得する。ここで必要になるのは各区分における値域の値と、その区分での一次近似に用いられる線分の傾きであり、前者に14ビット幅32ワードのメモリP(448ビット容量)と、後者に9ビット幅32ワードのメモリQ(288ビット容量)の、2つのメモリである。

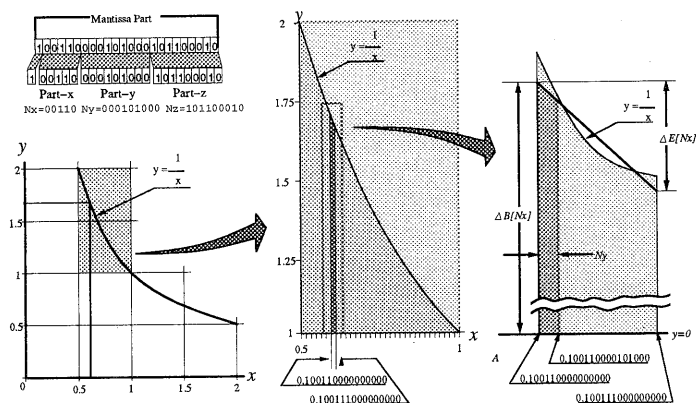


図3: N/R 法初期値獲得概念図

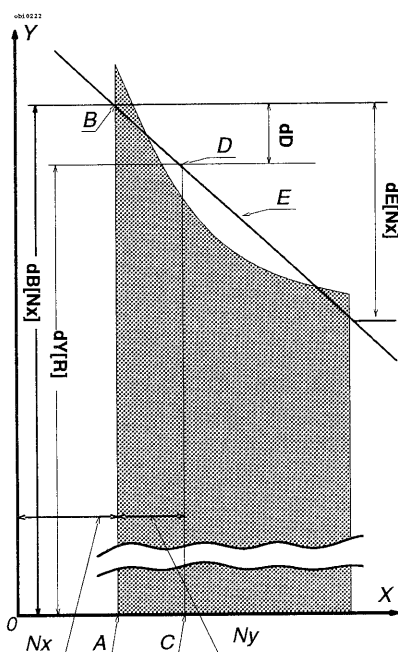


図4 N/R 法初期値獲得

併せて736ビット容量のメモリを用意すれば足り、FPGA上にも容易に実現可能である。一方で4入力のLUTはプログラミングに依って16ワードの小容量メモリとして使用出来るので、これを2つ接続して32ワードのメモリとして使用する。但し2個のLUTを論理的に接続するには更にもう一つのLUTを充てる必要があり、1ビット幅32ワードのメモリを構成するには3個のLUTが必要であるので、メモリとして用いるLUTは計69個となる。また、一次近似の為に9ビット×9ビットの乗算器(計90個のLUT)と、9ビットの加算器(実際には繰り上げも必要なの

R = 0.100110000101000101100010

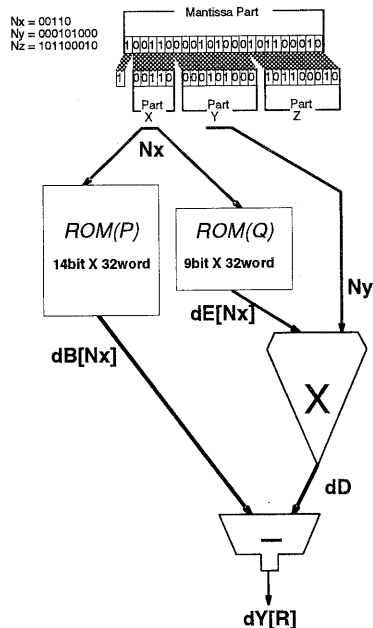


図5 N/R 法初期値計算

で14ビット,計15個のLUT)が必要となる。

結局一次近似を行なう手法を用いる事で計180個程のLUTがあれば初期値を得る事が出来る。実装するCPUの命令として、この演算を行なう回路を駆動する命令を追加し、以後、N/R法で逆数を求めてやれば良い。

3.3 初期値獲得の手順

まず被除数Rの仮数部を示すビット列の上位からMSBの1ビットをおいて、5ビットを取り区分Nxとおき、また引き続き4ビットを取り区分Nyとおく。

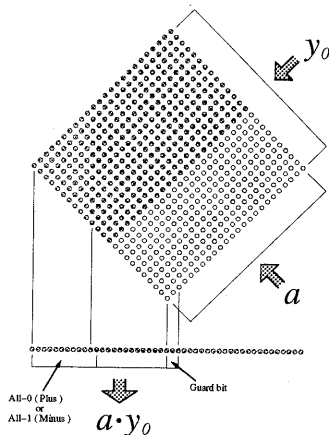


図6 $(a \times y_0)$ の計算

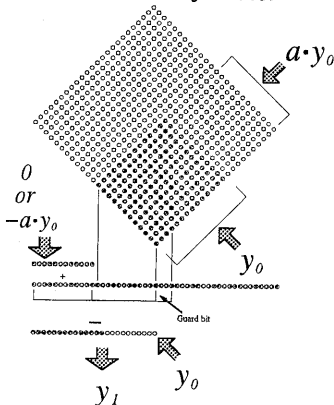


図7 $y_0 \times (1 - a \times y_0)$ の計算

次に区分 N_x の値を用いて与えられた除数に近い 32 分点 A を取り、それに対応する除数の逆数の近似値 B を得る(図 4)。これは 14bit $\times$ 32word の小容量 ROM(P)を用いて $\Delta B[N_x]$ として索く。同時にこの区分に於ける線分 E の傾き $\Delta E[N_x]$ を 9bit $\times$ 32word の小容量 ROM(Q)より索く。次に、区分 N_y より線分 E 中の 16 分点(定義域の中では 512 分点)から除数に近い点 C を取る(図 4)。この Y 軸座標上の影までの距離、つまり ΔY は、 $\Delta E[N_x]$ と区分 N_y を乗じて得た ΔD を $\Delta B[N_x]$ から差し引いて求められる。

3.4 N/R 法反復回路の実装

また、この逆数 a の初期値を Y_0 と於いて、24bit 精度の逆数 Y_1 を得るには、

$$\begin{aligned} Y_1 &= Y_0 \times (2 - a \times Y_0) \\ &= Y_0 + Y_0 \times (1 - a \times Y_0) \end{aligned}$$

を計算すれば良い。この際に、複数の演算命令を組み合わせて処理してもよいが、一回の反復に 3 命令を要し(定数 2.0 を定義する場合には更にもう 1 命令)パフォーマンスが低下する。これを解消するために反復演算そのものを回路として設計する。

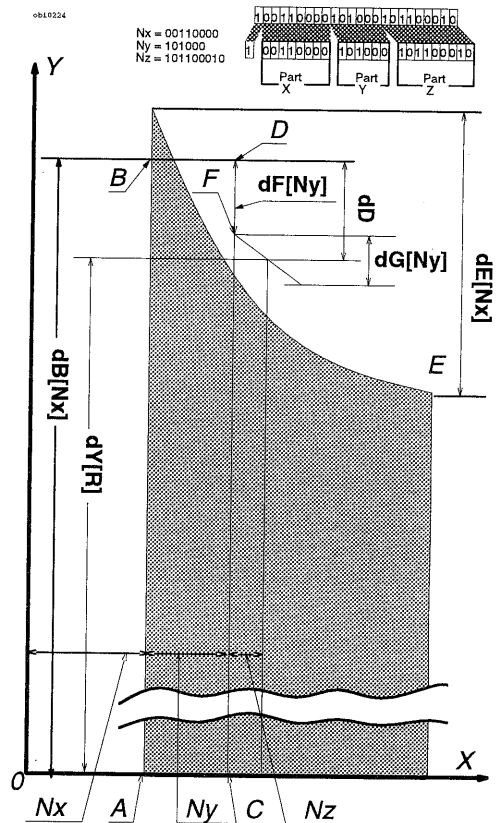


図8 直接逆数を求める

この際に、有効なビットについて注意すると回路資源を節約できる。即ち、初期値 Y_0 は精度が 14 ビット以下なので、MSB より 14 ビット分の演算を行えばよい。この場合の、 $a \times Y_0$ の乗算は図を行なえば良く、図 6 の濃色部分のみ計算すればよく、240 個の LUT を充てれば足りる。また $(1 - a \times Y_0)$ では上位の 10 ビットは全て 0 か、全て 1 になるので、この部分の演算も省略できる。引き続き、更に Y_0 を乗ずるが、この時にも無効なビットについての演を省略して、図 7 の濃色部分の計算について 150 個の LUT を充てれば足りる。

結局 1000 個程度の LUT があれば、単精度の逆数計算そのものが実装でき、乗算とほぼ同程度となる。

3.5 逆数の獲得

上記で述べた小容量の ROM と小規模な乗算を用いて与えられた除数の逆数を得る方法で、近似値を求める区分点数を増やし、より高い精度で近似値を求める、即ち最初から 24bit 精度の逆数値を求める事も考えられる。然しこの精度を得るには区分 X として 10 ビット以上が必要であり、当然 LUT を組み合わせて算出する事は事実上不可能であり FPGA のメモリ機能を充てても不経済である。そこで、上記では各区分において直線で近似していたものに代えて、曲線で近似する。即ち、曲線を出力する小型の ROM を用意して、これで区分内の曲線を近似する。

例えば区分 N_x として 8 ビットを、続いて区分 N_y として 6 ビットを、最後に区分 N_z として 9 ビットを取る。区分 N_x より最初に 256 分点を定めて各点に於ける逆数の近似値 $\Delta B(N_x)$ 、並びにその区間の大まかな傾き $\Delta E(N_x)$ を索き、256 区間の中の各区間で区分 N_y より 64 分点を取り(全体の中では 16384 分点)、その中での補間の近似値 $\Delta F(N_y)$ 並びに傾き $\Delta G(N_y)$ と補正值 $\Delta H(N_x)$ を索く。この 5 個の数値と区分 N_z の値から、任意の除数の逆数の近似値の Y 軸座標上の影までの距離、つまり ΔY は、

$$\Delta Y = -(\Delta F(N_y) + (\Delta G(N_y) + \Delta H(N_y) \times N_x) \times N_z) \times \Delta E(N_x) + \Delta B(N_x)$$

で求められる。 ΔY は除数の逆数の近似値ではあるが区分 N_x に 8 ビット以上、区分 N_y に 6 ビット以上を確保する事で ΔY を 24 ビット前後の精度で求める事が出来る。即ち除数の逆数をほぼ 1 回の操作で求める事が可能である(図 8、但し ΔH の項は図から省略)。尚、 $\Delta H(N_y) \times N_x$ 、 $(-) \times N_z$ の演算には 9bit $\times$ 9bit の乗算器、同じく $\Delta E(N_x)$ を乗ずる部分には 16bit $\times$ 16bit の乗算器を用いれば良い。

4. 各演算方式の比較

図 9 に各演算方式のタイミングの比較を示す。ここでは、ALTERA 社の 10K シリーズを前提とし、一つの LUT 乃至メモリから次の LUT 乃至メモリへの遅延が 10ns、また加算時のキャリーの遅延は 1 段あたり 0.5ns と仮定する。図中の数値は遅延時間を ns で示す。

(A)は、従来の 1 ビットずつ商を算出する方式である。(B)は基数を 4 とした場合で 2 ビットずつ商を算出する場合である。(C)は SRT 法の場合で、LSB の不確実性を解消するために余計に 1 回反復を行う。尚、SRT 法では表のエントリが 11 ビットで表現されるために、FPGA のメモリ機能を使用する必要がある。この場合にはメモリ機能の制限から、完全なパイプライン構成にすることはできず、数段のみ実装して反復使用することになる。(D)は N/R 法に、反復専用回路を組み合わせた場合である。(E)は、3.5 項で述べた、直接単精度で逆数を求める場合である。

ここから、FPGA 上に逆数計算回路を実装するには、速度的には N/R 法と反復計算回路を直接実装する方式と、直接逆数を算出する方法では時間的には拮抗していると思われる。

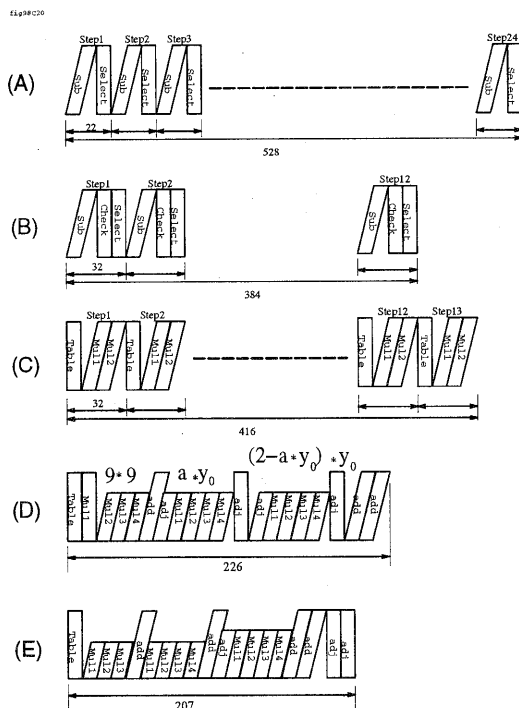


図8 各種方式の遅延

使用する回路資源の面からは、直接逆数を算出する方式では FPGA のメモリ機能を使用するため、効率が悪い。

5. まとめ

FPGA ベースのハードウェアアーキテクチャエミュレータに欠けていた浮動小数点除算命令を効率的にサポートする逆数計算回路を設計した。従来の 1 ビットずつ商を得る方式では、プログラムで命令を組み合わせる必要があり、マイクロプログラムで処理する必要があるが、同一回路を反復使用する場合に資源の衝突が起こるなどの難点があった。今回設計した N/R 法と反復計算の組み合わせの回路、あるいは直接単精度の逆数を計算する回路では、占有する回路資源がかなり増大するが、乗算回路程度に抑えられるので、容易に FPGA 上に実装できる。また、完全にパイプライン構成をとることが出来るので処理効率も良い。

今後は設計した回路の検証、特に誤差の検証を行う。また、IEEE-754 規格の丸め処理等も追加し、同規格との互換性を取る。

参考文献

- [1] D.Patterson, J.Hennessy, "Computer Architecture A Quantitative Approach", Morgan Kaufmann Publishers, inc., 1990.
- [2] D.E.Atkins: "Higher-Radix Division Using Estimates of the Divisor and Partial Remainders", IEEE Trans. Computers, vol.17, p.930, Oct., 1968
- [3] A.D.Booth: "A Signed Binary Multiplication Technique," Quarterly J.Mechanics and Applied Mathematics, Vol. 4, Part 2, 1951.
- [4] C.S. Wallace: "A Suggestion for a Fast Multiplier," IEEE Trans. Electronic Computer, vol. EC-13, pp.14-17, Feb. 1964
- [5] D. Ferrari: "A Division Method Using a Parallel Multiplier," IEEE Trans. Electronic Computer, vol. EC-16, Feb. 1967
- [6] 柏倉 正一郎, 井上 淳樹, 大江 良一, 御手洗 伸, 鶴隆行, 伊澤 哲夫, 後藤 源助: "符号選択式 Booth Encoder を用いた 54b×54b コンパクト乗算器," 信学技報 ICD97-3, CPSY97-3, FTS97-3, Apr. 1997
- [7] クアメ・オセイ・ボアテン, 高橋 寛, 高松 雄三: "改良 Booth 法に基づく乗算回路の C-テストブル設計について," 信学技報 ICD97-3, CPSY97-3, FTS97-3, Apr. 1997
- [8] T.M.Conte, C.E.Gimarc, "Fast Simulation Of Computer Architectures," Kluwer Academic Publishers, 1995
- [9] 尾形, 山本, 水尾, 木村, 笠原: "FPGA を用いたマルチプロセッサシステムテストベッドの実装," 情報処理学会 ARC-128-14, pp.79-84, 1997