section 5

私のディジタル回路設計ノート

高精度の逆数(1/x)を計算する

畔津明仁

これまでの本連載では，数値演算にまつわる話題のうち，入出力のビット数が32ビット以下のものを取りあげてきました．これを演算するかテーブルにするか，あるいは誤差や単調性をどう考えるかについて，例をあげました．これらは従来のシステム(アナログ/ディジタルを問わず)で高速を要求される分野で役立つ例です．今回と次回とは，これまでと少し趣が異なります．それは速度より精度を要求される場面です．そうはいっても，ハードウェア設計者の出番は，ソフトウェアより1桁以上の性能(処理速度)アップですから，一瞬(1psまたは1ゲートというべきか？)たりとも気は抜けません．

1．逆数という演算

◆なぜ1/xか？

この連載の1回目(1999年12月号)で割り算を取りあげました．そこでも触れましたが，逆数があれば割り算ができるし(掛け算は必要)，逆に割り算があれば逆数は得られます(図1)．

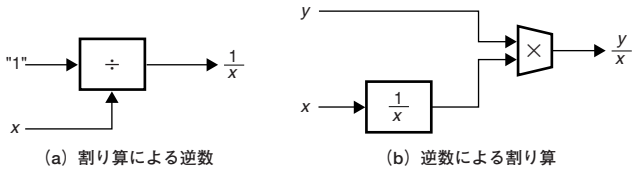
一般の用途では，逆数($1/x$)が単独で必要な場合も少なくありませんが，割り算(y/x)の頻度のほうが多いでしょう．これは第1回でも述べたように，割り算が数学上の“体”(field：たとえば実数の集合は“体”である)を作るための基本演算の一つであるだけでなく，私たちの身の回りに有理式[注1]で表現されるものが非常に多いことから明らかです．すでに述べたように，割り算が必要であって，しかも専用回路化にメリットがあるのなら，躊躇(ちゅうちょ)なく割り算回路を作るべきです．回路設計者の存在価値はまさにこの点にあり，割り算が必要とされたときに「できません」では許されません．

一方，システムの用途によっては，割り算(y/x)の代わりに逆数($1/x$)でもよい(あるいはメリットがある)こともあります．たとえば，データ転送レートに比べて割り算の出現頻度が低いとか，汎用演算としての掛け算が容易に使えるとかの条件がある場合です．

このような場合，割り算回路の代わりに逆数回路をもつことにより，

(1)回路規模(ゲート数)が削減できる．
(2)割り算回路(入力2ポート)に比べて入力信号線数が半減する．これはテスト上も好都合である．
(3)割り算より逆数のほうがアルゴリズムの選択肢が多い．

といった長所も出てきます．



(a) 割り算による逆数　　(b) 逆数による割り算

〔図1〕逆数と割り算

◆高精度演算

これらの長所は，演算精度(ビット長)が長い場合には重要です．第1回でとりあげた32ビット浮動小数点形式(IEEE 754形式)では，仮数部が24ビット[注2]なので，入力が2ポートになっても割り算回路(仮数部)としては，たかが入力48ビットにすぎません．

しかし，80ビット浮動小数点形式(仮数部64ビット)では，2ポートの入力は128ビットです．1個のサブモジュールとして，64ビット入力と128ビット入力とでは設計はもちろん，シミュレーションやテストに必要な手間が雲泥の差であるのは，容易におわかりいただけるものと思います．

今回お話しするのは，ソフトウェアに比べてハードウェアが不得手とする高精度演算の問題点と，ソフトウェア/ハードウェアで同じような数学的アルゴリズムを使いながらも実現手法が異なるという点です．

2．1/xのアルゴリズム

◆高精度演算のむずかしさ

符号付き数値の場合，16ビットの1LSBはF.S.(フルスケ

注1：g/fあるいは$\Sigma(g_n/f_n)$の形で表される関数(f，gなど整次多項式)．第1回ではその一例として同次座標変換$(ax+by+cz+d)/(ex+fy+gz+h)$をあげた．
注2：MSB(いわゆるインプリシット・ビット)を含める．2進数で正規化を行うと，仮数部の範囲は，
1▲000....0 ～ 1▲111....1　　(▲は，小数点の位置を示す)
となるのでMSB(整数部1桁目)は‘1’に固定となる．この‘1’を省略したのがインプリシット・ビットで，この表現形式を「ケチ表現」という．

ール）× 2^{-15}，つまり0.003%（=30ppm）です．わたしたちは数値を扱うかぎり，アナログ的思考をしますが，現実のアナログ回路で30ppmの精度を保証するのはなみたいていのことではありません．

時間や周波数のうえでの30ppmはギリギリなんとかなるとしても，電圧や電流レベルでの30ppmは実回路上つらいものです．

ディジタル回路が全盛の世の中となって，そのビット数が平然と語られるようになりました．たしかに，ディジタル録音のビット数として“16”は“12”より上です．ただし，聴感効果が16倍（$2^{16}/2^{12}$）ではないはずです．まして，“16ビットの絶対精度”が簡単に得られるか否かは，回路設計をしたことのある読者なら簡単に判断できると思います．

しかし，ディジタル回路の精度（分解能）として，16ビットを越える要求はザラです．32ビット浮動小数点の仮数部の1LSBは2^{-23}，つまり0.1ppmです．80ビット浮動小数点数では仮数部は64ビットであり，1LSBはppb（10^{-63}）以下です．

これはアナログ回路ではほぼ不可能な精度です．また，ディジタル（ソフトウェア/ハードウェアを問わず）では，資源と手間とを費やせば可能ではあるのですが，性能（速度）と経済性とを問われます．そして，忘れられることが多いのですが，精度の検証や回路（またはソフトウェア）のテストをどうやって行うかも重大なポイントです．

◆高速化の要求

1990～91年にかけて，筆者は長語長（浮動小数点数で64～80ビット）の演算を高速化したいという要求を複数の依頼先から受けました．演算の種類としては四則演算および初等関数が考えられました．

そのころの演算速度としては，単精度（32ビット）浮動小数点の加減乗算でレイテンシ50ns～100nsのFPUがありました．倍精度ではこの数倍が必要で，かつ割り算や逆数では，そのまた数倍となります．結局，倍精度以上の割り算は1 μs前後のレイテンシが必要でした．

表1に示すのは，米国MIPS社の資料（1991年）の一部です．これはアーキテクチャの解説であり，実際の時間は記されていません（ハードウェア=CPUの動作周波数に依存する）が，このころのRISC CPUの多くは30MHz～40MHzの動作周波数だったと記憶しています．だとすれば，割り算では1 μs，開平では3 μs程度のレイテンシだったことになります．

いくつかの依頼のうち，普通の倍精度演算（たとえばIEEE754の64ビット形式）のものは，受注を断りました．これは市販LSI（FPUやCPU）がサポートし続けるであろうし，その性能も順調に向上するに違いないことが理由の一つでした．

〔表1〕MIPS社のR4000のFPU

演算	レイテシン（パイプライン・サイクル数）	
	単精度	倍精度
ADD，SUB	4	4
MUL	7	8
DIV	23	36
SQRT	54	112

〔MIPS社“R4000 MIPS Microprocessor User's Manual”（1991年）掲載のFloating-Point Operation Latencies より一部引用〕

一方，80ビットの拡張形式については，技術的な興味を捨て切れませんでした．一部のコプロセッサ（たとえば8087）に80ビット形式で演算ができるものがありましたが，性能・機能ともに不十分だったためです．つまり，「ハードウェア化の意義がある」と思えたのです[注3]．

◆割り算と逆数

次に考えたのは，このような長語長の演算を高速化するために「どのようなモジュールを作成すべきか」でした．

もちろん，加減算をはじめとして必要な演算モジュールをひととおり新規作成すれば，高性能は達成できるでしょう．しかし，それは人手のむだになります．80ビット形式の演算ハードウェアが世の中に少ないとはいっても，加減算や乗算なら皆無でなく，またその気になればだれでも設計できます．ここで手間をかけて機能を競っても，差はわずかなものです．

一方，割り算や初等関数はどうでしょう．**表1**からもわかるように，割り算や開平は加減乗算の数倍から10数倍の演算時間が必要です．また，依頼元からは三角関数の使用頻度も多いという話がありました．そこで，筆者らは，加減乗算以外の目的のモジュールだけを新規に作成したいと考えました．

今回の話題の割り算については，あえて逆数回路を作成することにしました．前述のように，逆数回路（入力1ポート）のほうがアルゴリズム選択肢が多く，高速化のチューニングも容易だからです．そしてなにより，乗算回路は他企業の努力（?）で高速化されるはずで，その成果を利用しない手はないと思いました．

◆逆数のアルゴリズム

割り算や逆数のアルゴリズムは，第1回（1999年12月号）でも触れましたが，その中の典型的なものを復習しておきましょう．

⑴筆算手順型：1999年12月号の**図12**のもの．
⑵関数発生型：第2回以降にたびたび登場するもので，細分した区間ごとに簡単な関数による近似を行うもの．
⑶漸近近似型：ニュートン・ラプソン法のように，誤差を補正して近似値を求めるもの（**図2**）．

もちろん，⑴，⑵，⑶以外にもさまざまなアルゴリズムが

注3：内輪話だが，管理上の反対を押し切って技術的観点から検討を継続した結果，大幅な納期遅れを出して依頼側に迷惑をかけてしまった．