割り算回路設計，あの手この手

連載 4

漸近法を用いた除算回路

鈴木昌治

デバイスの記事

関連データ

今回は，乗算をベースにする漸近法を用いた除算回路の実現方法を解説する．具体的には，除算に特化した「反復乗算法」と汎用的な「Newton-Raphson法」の二つのアルゴリズムを用いた回路設計を紹介する．なお，本稿で設計した回路のソース・コードは本誌ホームページ（http://www.cqpub.co.jp/dwm/）からダウンロードできる．（編集部）

これまで3回にわたって，回復法や非回復法を用いた除算回路設計および高速化について話をしてきました．これらは，いずれも「被除数の中に除数がいくつ取れるか」という視点に立ったものでした．

今回は，「被除数と除数の比を求める」という視点で除算を実行する方法について考えてみましょう．代表例として，漸近法を取り上げます．

1 漸近法とは

たしか高校の数学だったと思いますが，漸化式というものを習ったと思います．初期値，およびi番目と$i+1$番目の変数の関係を定義することで，すべての変数を定義できるというものです．$i \to \infty$としたときに収束するか，なども議論されていたように記憶しています．

漸近法はこの考えかたを応用したものです．初期値からはじめてi番目から$i+1$番目の変数を求めることを繰り返し，特定の値に収束させることで，関数の近似を行うというものです（もちろん，無限にサイクルを繰り返すのは現実的ではないため，有限回のサイクル後に必要な精度を得るものとする）．漸近法の中には，例えばNewton-Raphson法（詳細は後述）に代表される汎用性の高いアルゴリズムもありますが，適用する関数によってくふうが凝らされたものもあります．まずは，除算に特化したアルゴリズムを紹介しましょう．

● 回復法も非回復法も実は漸近法の一種

ところで，回復法を用いた除算回路では次の式を繰り返し演算していたことを思い出してください．

$$\left.\begin{aligned} pq_i &= 1 \\ pr_{i+1} &= pr_i - d \end{aligned}\right\} \quad (pr_i \geqq d \text{の場合}) \cdots\cdots (1\text{-}1)$$

$$\left.\begin{aligned} pq_i &= 0 \\ pr_{i+1} &= pr_i \end{aligned}\right\} \quad (pr_i < d \text{の場合}) \cdots\cdots (1\text{-}2)$$

ここで，pr_i：部分余，pq_i：部分商，d：除数

実は，条件分岐があるとはいえ，回復法や非回復法なども漸近法の一種なのです．けた単位での漸近を反復することで，商を必要とする精度に追い込んでいきます．しかし，手段に筆算のイメージが強いためか，あるいは結果が商と余で誤差なしで与えられるためか，漸近法として語られることはないようです．

● 乗算ベースの漸近法

回復法や非回復法は，けた単位で加算（減算）を行う方法でした．つまり，サイクル当たり1ビットの商を求めようとすれば，kビットの結果を求めるのにkサイクル必要となるのです．

これに対し，今回は乗算をベースにすることで指数的に精度を上げられる方法を考えてみましょう．ここでは，「反復乗算法（あるいは，Goldschmidtのアルゴリズム）」と言われる方法を用いることで，反復するサイクルを$\log_2 k$に短縮できることを説明します．

商をq，被除数をz，除数をdとすると，$q = z/d$となり

ます．つまり，求めるものはzとdの比となり，式(2)の表現を用います．

$$q=\frac{z}{d}=\frac{z}{d}\cdot\frac{\prod_{i=0}^{k-1}x_i}{\prod_{i=0}^{k-1}x_i} \quad \cdots\cdots(2)$$

まず，この式が何を表しているかを説明します．これは，「iサイクル目でx_iという変数を分母と分子それぞれに乗算する」という操作を反復するものと考えてください．分母と分子に同一のものを乗算しているので，qとしては変化しないことに着目しましょう．ここで，iサイクル目における分母をd_i，分子をz_iとすると，式(3)が成立します．

$$\left.\begin{aligned} d_{i+1}&=d_i\cdot x_i\\ z_{i+1}&=z_i\cdot x_i \end{aligned}\right\} \quad \cdots\cdots(3)$$

これが漸化式となり，初期値はdとzです．x_iを乗算する操作を反復することで$d_i\rightarrow 1$という漸近操作ができれば，$z_i\rightarrow q$という結果を得ることになります．つまり，分母を1に追い込んでいくことで，自動的に分子がqに追い込まれるというわけです．

では，x_iをどのように定義するかを説明しましょう．まず，$d_i\rightarrow 1$を実現することは$1-d_i\rightarrow 0$を実現することと等価と考えることができます．式(2)の分子の初期値は，

$$1/2\leqq d<1 \quad \cdots\cdots(4)$$

に正規化しておきます．すると，

$$0<1-d\leqq 1/2 \quad \cdots\cdots(5)$$

となります．このため，$1-d$のべき乗であれば0に近づけることが可能であると判断できます．ここでは乗算を用いて漸近操作を行いますが，次数を高くとると乗算回数が多くなります．そこで，次数は2次（乗算1回）として，以下のように漸化式を定義しましょう．

$$1-d_{i+1}=(1-d_i)^2 \quad \cdots\cdots(6)$$

式(6)を変形すると，

$$1-d_{i+1}=(1-d)^{2^i} \quad \cdots\cdots(7)$$

となり，サイクルが進むほど$1-d_i\rightarrow 0$を実現できることが確認されました．また，式(6)は次式のように変形できます．

$$d_{i+1}=d_i(2-d_i) \quad \cdots\cdots(8)$$

式(8)に式(3)を代入すると，式(9)が得られます．

$$x_i=2-d_i \quad \cdots\cdots(9)$$

これを順次d_i，z_iに乗算していくことで，$d_i\rightarrow 1$，$z_i\rightarrow q$が実現できます．

● 誤差があることを考慮して変数のビット幅を決める

演算回路（ハードウェア）における変数のビット幅をkとすると，d_iは$1-2^{-k}$以上には1に近づくことができません．したがって，$2^i\geqq k$を満たすiに至った時点で演算の反復動作は終了となります．このことから，反復回数iは，次の式で与えられます．

$$i\geqq\log_2 k \quad \text{を満たす最小の整数} \quad \cdots\cdots(10)$$

回復法や非回復法などの場合，除数，被除数，商，余の関係には誤差がありませんでした．一方，反復乗算法では「得られたq（商）は基本的に誤差を含んでいる」ということに注意しなくてはなりません．演算回路に誤差がなかったとしても，演算方法そのものに「d_iは$1-2^{-k}$以上には1に近づくことができない」という大前提があるので，分子のqもその分小さな値をとることになるからです．

また，乗算を行った結果，変数のビット幅は$2k$となります．これを次のサイクルにそのまま持ち込むと，変数の幅がサイクルを重ねるたびに大きくなってしまい，現実的ではありません．乗算結果はkビットに丸めて，次のサイクルに渡すのが一般的でしょう．この丸めによっても誤差が生じることになります．丸めによる誤差をεとすると，誤差を含んだd_i'は式(11)のように表現できます．

$$d_i'=d_i+\varepsilon \quad \cdots\cdots(11)$$

式(11)を用いて式(3)を書き直すと，式(12)が得られます．

$$\left.\begin{aligned} d_{i+1}'&=(d_i+\varepsilon)(2-d_i-\varepsilon)\\ &=d_i(2-d_i)+2\varepsilon(1-d_i)+\varepsilon^2\\ z_{i+1}'&=z_i(2-d_i-\varepsilon)\\ &=z_i(2-d_i)+\varepsilon z_i \end{aligned}\right\} \quad \cdots\cdots(12)$$

丸めを単純な切り捨てで行ったとすると，$\varepsilon<1\text{ulp}$（unit in least position；最下位ビット）となります．ε^2は十分小