割り算回路設計，あの手この手

連載 2

デバイスの記事

関連データ

非回復法による除算回路の設計

鈴木昌治

今回は，非回復法というアルゴリズムを用いた除算回路の設計法を説明する．回復法とは異なり，非回復法では部分余が除数より小さい場合でも減算を行う．その結果，部分余が負になった場合は次のサイクルで除数を加算する．つまり，符号付きの演算が行える．なお，本稿で紹介した回路のHDLソースは，本誌付属のCD-ROMに収録されている．（編集部）

前回（本誌2005年8月号，pp.109-117）は回復法をベースとして除算回路の基本についてお話ししました．そのとき紹介した回路はあくまでも基本的なものであり，動作速度の改善や回路規模の抑制などに不利な面があり，また無符号の引き数しか扱えない構造でした．

今回は非回復法を取り上げます．ここでは，取り扱う引き数も符号付きへと発展させ，高速化へのアプローチとしましょう．

1 引きすぎてもよい「非回復法」

非回復法は，基本的な構造そのものは回復法と似ており，上位側から商を確定していき，徐々に余を0に近づけていくという方法です．もっとも大きな違いは，「引きすぎてはならない」という回復法特有の制約を取り払っていることです．回復法に見られた筆算の延長上では，ある意味「非常識」なことなのですが，どうしてそのようなことが可能なのでしょうか．

●「部分余が負になったら加算」というルールを追加

回復法において，iサイクル目でシフト・アップ済みの部分余をpr_iとします．このサイクルで減算の条件（$pr_i \geqq d$[注1]）が満たされれば，式(1)が実行されます．

$$sub\_result_j = pr_j - d \quad \cdots\cdots(1)$$

逆に，減算の条件が満たされなかったとすると，減算は実行されず（回復），pr_iは次のサイクルに持ち越しとなります．次のサイクルで減算が実行されると，式(2)が実行されます．

$$sub\_result_{j+1} = 2pr_j - d \quad \cdots\cdots(2)$$

この2サイクルにおける商は，$01_2 = 0.5_{10}$です．

ここで，減算の条件が満たされていないにもかかわらず，iサイクル目で減算を実行してしまった（非回復）としましょう．すると，次のサイクルで減算を実行した場合に，

$$\begin{aligned} sub\_result_{j+1} &= 2\left(pr_j - d\right) - d \\ &= 2pr_j - 3d \end{aligned} \quad \cdots\cdots(3)$$

となり，式(2)とは異なる誤った結果となります．

このとき，加算が可能であるとしたらどうでしょうか．つまり，減算の条件が満たされていないということは式(1)を実行すると負になりますが，そこに「部分余が負のときは除数を加算する」というルールを適用するわけです．このルールを適用したものが式(4)であり，答えも式(2)と一致します．

$$\begin{aligned} sub\_result_{j+1} &= 2\left(pr_j - d\right) + d \\ &= 2pr_j - d \end{aligned} \quad \cdots\cdots(4)$$

このとき部分商としては-1を立てて，加算という逆の

注1：本連載において，dは除数を，zは被除数を，qは商を，rは余を表すものとする．

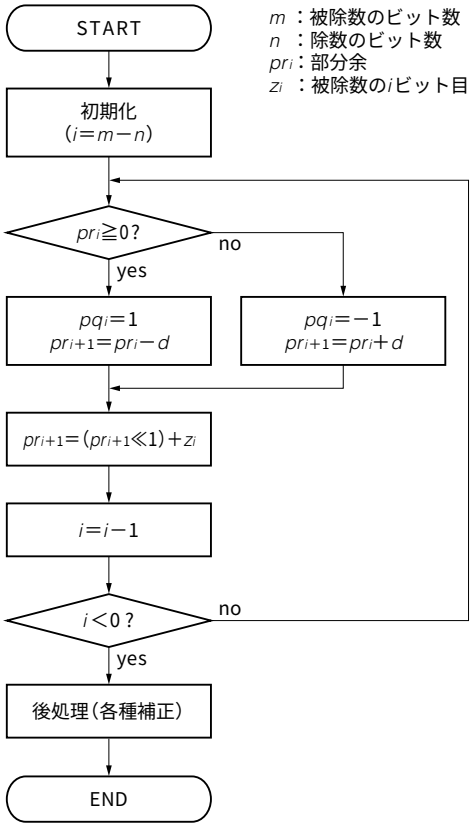
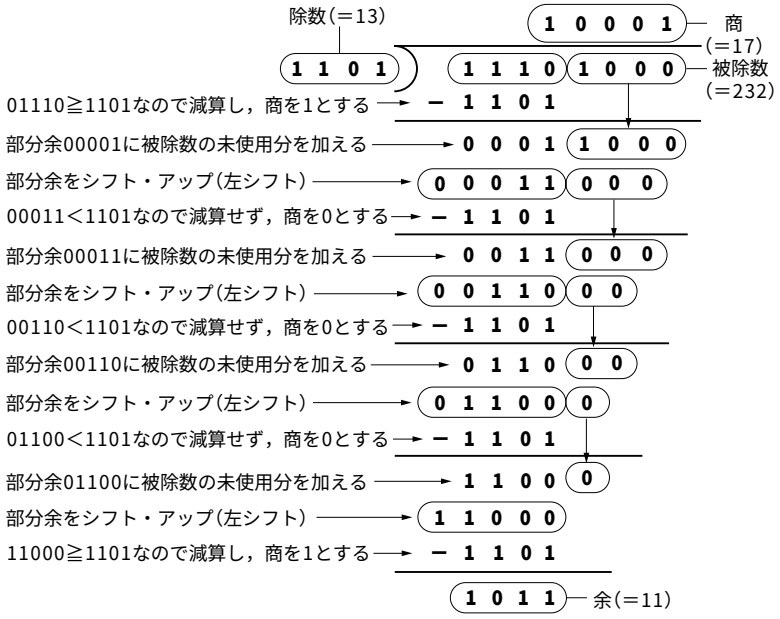
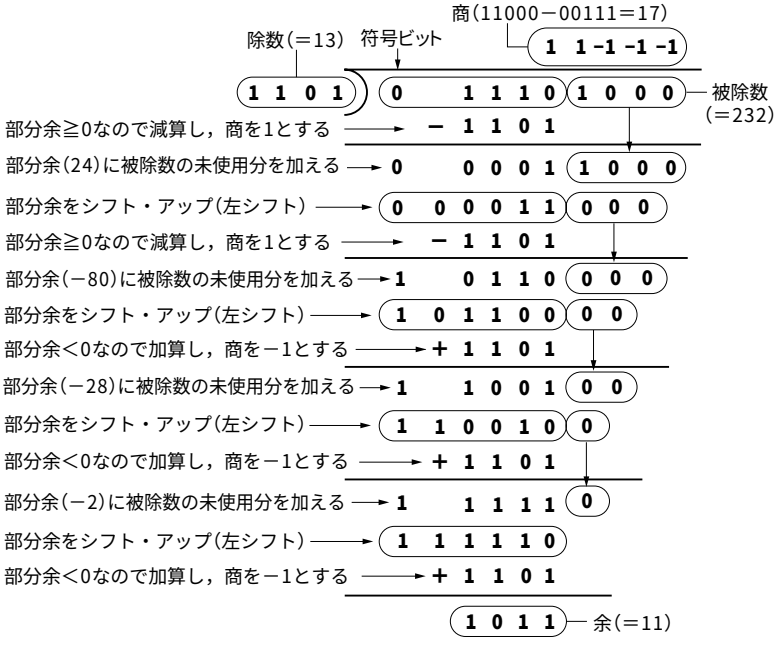


図1　非回復法による除算アルゴリズム

非回復法による除算アルゴリズムをフローチャートで示す．



(**a**) 回復法による除算例



(**b**) 非回復法による除算例

▶ 図2
除算の例

比較のため，(a)に回復法による除算を示す．(b)の非回復法では，回復法の場合と異なり，部分余が一時的に負の値をとることに注意．なお，前回は，算数で習った筆算の延長で説明したため，けたを不動のものとして扱ったが，論理回路を考えるうえでは図のように左詰めにするほうがより現実的といえる．

操作を実行した結果を残します．この2サイクルにおける商は，$(10-01)_2=(1-0.5)_{10}=0.5_{10}$で，商としても一致していることがわかります．

こうして，回復法における「iサイクル目の減算のスキップと$i+1$サイクル目の減算実行」で得られる結果と同じものが，「iサイクル目の減算実行と$i+1$サイクル目の加算実行」で得られたわけです．これが非回復法です．iサイクル目における誤った(？)減算を$i+1$サイクル目で加算することで回復しているので，「非回復」というのもおかしいのですが，回復のステップが1サイクルで閉じていない，というところが回復法との大きな違いといえるでしょう．

● 操作は異なるが回復法と同じ結果が得られる

フローチャートを用いた非回復法のアルゴリズムを**図1**に示します．また，**図2**にその例を示します．ここで，一つ注意が必要です．それは，非回復法では部分余が一時的