アルゴリズム 29－5
（1992. 8. 25）

スケーリング操作により商数字選択を簡単にした基数2の除算器

外村元伸：（株）日立製作所中央研究所

本報告は，部分剰余計算に｛－1，0，＋1｝で表現する桁上げ伝播の生じない冗長2進加算器を利用して基数2の効率的な除算器を構成することに関するものである．従来のように除数を［1／2，1）の区間に正規化する方法では，商数字を選択するためには部分剰余値の3桁を参照する必要がある．ところが，除数の範囲を［1，3／2）の区間にずらすと（これをスケーリングと呼ぶ），部分剰余値の2桁を参照するだけで商数字が選択でき，1桁の商数字選択回路につき論理段数にして4段から2段へと，従来よりも半減できる効率的な除算器が構成できることを示す．これには，外村の提案した演算アルゴリズムの高階ストリング・グラフ表現の手法［22］を応用して導出する．

Simple Quotient-digit-selection Radix-2 Divider with Scaling Operation

Motonobu Tonomura,
Hitachi Central Research Laboratory, Hitachi Ltd.

This paper deals with a theory and design method for an efficient radix-2 divider using carry-propagation free adders based on redundant binary {-1,0,+1} representation. In the usual method of normalizing the divisor in the range [1/2, 1), it is necessary for selecting a quotient digit to refer to three significant digits of the partial remainder. However, here we show that it is possible to select a quotient digit to refer to only two significant digits of the partial remainder, scaling the divisor in the range [1, 3/2). So, the divider becomes half of the logic depth (four in usual methods) for each quotient digit selection circuit and it can be designed more effectively. This was derived from an application for the hyperstring graph representation proposed by M. Tonomura for arithmetic algorithms [22].

1．はじめに

一般に，乗算アルゴリズムでは，すべての部分積を同時に得ることができる．しかし，除算アルゴリズムでは，部分剰余が関与するため前後の処理が独立でなくなり，順序性がある．そのため，逐次的に部分商を求めなければならない．そして，長いあいだ並列除算法は知られていなかったが，1980年代半ば頃に，P. W. Beame ら［3］によって内部計算に剰余数表現を用いる並列計算法が明らかにされた．しかし，岡部寿男ら［14,15］の具体的な考察によれば現実的な素子数規模ではまだ実現できない．現在までに様々な除算アルゴリズムが考案されているが，処理ビット数が３２ないし６４では，Newton 法などにより除数の逆数を求め，除数の逆数と被除数を乗算する方法，あるいは逐次処理法を基本にしたものなどが実用的である［12,18,24］．逐次型除算アルゴリズムを高速化する１つの方法として，２より高い基数を導入することが考えられている［1,2,8,9,11,21］．従来から研究されているのは，ほとんどが桁上げ保存加算器を利用したものである［12,21,25］．それに対して，高木ら［17,19］は｛－１，０，＋１｝の冗長２進数表現にもとづいた冗長２加算器を利用した基数２の除算方式を提案し，谷口・枝末・西山・國信［5,13,20］によって実現された．一般に，冗長２進の場合の方が，桁上げ保存の場合に比べて負の数の扱いが簡単なためその構成が容易になる．M.D. Ercegova［8］は，冗長２進加算器を利用した基数４の方式を提案している．彼の方法の注目すべき点は，従来のように［１／２，１）の区間に正規化する方法では商数字の選択に必要な桁数が多すぎて基数４以上を使う利点が得られないことから，正規化区間より若干ずれた［６３／６４，９／８）の区間に収めるスケーリング法というものを提案し，商数字選択の効率を高めたことである．本研究でも彼のスケーリング操作の概念を別のアプローチから支持できる結果を得た．高基数の場合を議論する前に，まず，基本として，基数２の場合を再検討して詳しく調べたので報告する．この検討のために，外村［22］が提案した演算アルゴリズムの高階ストリング・グラフ表現による解析法を応用する．商数字の選択には，部分剰余値の３桁以上を参照するものをほとんどの研究者が採用している．２桁のみの参照で可能なことは知られていなかったが，Svoboda のアルゴリズム[1963年:16,23]を適用すれば得られることを，最近，N. Burgess [4]が示した．本研究でも，部分剰余値の２桁のみの参照で商数字が選択できることをスケーリング操作と独自の解法で再確認した．

2．高階ストリング・グラフ表現

この章では，まず，ストリング・グラフ表現の定義を行い，それから本除算方式で利用する冗長２進加算器の演算規則を扱うための高階ストリング・グラフ表現について説明する．

Aを空でない記号の有限集合とするとき，Aの要素の有限記号列はA上のストリングと呼ばれる．n>0 について，A^nはA上の長さnのストリングすべての集合を表す．また，A上の空も含むすべてのストリングの集合は，A^*によって表す．a,b∈A^*に対して，ab は２つのストリングaとbの連結ストリングを表す．任意のa∈Aとx∈A^*に対して，演算子∂rと∂lを

$$\partial r(ax) = \partial l(xa) = x \qquad (2.1)$$

によって定義する．ストリング・グラフは４項組 S = (A^{n-1}, f, A^n, ζ)によって定義される．ここで，fは任意のx∈A^n に対して，

$$f(x) = (\partial r(x), \partial l(x)) \qquad (2.2)$$

のようにA^{n-1}の要素対で表される(図2.1)．A^{n-1}の要素をストリング・ノード，A^nの要素をストリング・ブロック（あるいは辺）と呼ぶ．f(x)には，∂r(x)から∂l(x)への向きが定義できるので，f(x)は有向ブロック（辺）であるという．ζは必要ならばA^{n-1}またはA^nの要素への意味付けを与えるラベル付け関数である．必要がないときには明記しない．例として，任意の２進数を下位桁から上位桁に向かって表すために図2.2に示すようにストリング・グラフS＝(B^2, f ,B^3)をつくる．ここで，B={0,1}である．この例ではラベル付けによって特に意味を与えないのでラベル付け関数ζがない．有向ブロックf(x)に関しては，例えば，２進数のある部分がx=011とすると，∂r(x)=11，∂l(x)=01であるから，ノード11からノード01への有向ブロック(11,10)で表現できる．また，それらのノード間の対応がはっきりしているので，ブロックの途中部分を図では共有させている（場合によっては省略してもよい）．その方が図を書く上では便利である．明らかに，ストリング・グラフSから有向ブロック列を生成することによって，任意の２進数を表現できることがわかる．

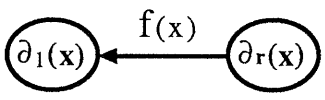

図２．１　ストリング・グラフの定義．

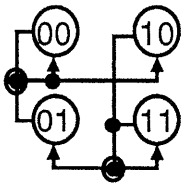

図2．2　2進数とその桁移動を表すストリング・グラフ表現.

$S = (B^2, f, B^3), B = \{0,1\}$

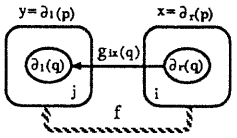

図2．3　加算を表現する高階ストリング・グラフの構成.

次に，$\bar{1}$=−1とするとき，$\{\bar{1}, 0, 1\}$で表現される冗長2進数の加算規則をストリング・グラフで表現する方法について説明する．ただし，本除算方式で採用する冗長2進加算器は，加数の方は{0,1}表現に制限し，被加数の方に$\{\bar{1}, 0, 1\}$表現を許すものである．既に説明したように，ストリング・グラフ$S=(B^2, f, B^3)$で任意の加数が表現できる．そこで，図2.3に示すように，加数のノードの中にさらに被加数のノードを定義することによってストリング・グラフを高階化し，被加数と加数の加算を表現する。また，加算では桁上げ状態が生じるので，桁上げ状態に応じて別々の高階ノードを定義する．すなわち，冗長2進加算規則の高階ストリング・グラフが，加算の桁上げ入力状態(キャリー) $i\in\{\bar{1}, 0, 1\}$および$x\in B^2$に対して，

$$HS=(\{Bix^2\}, \{gix\}, B^3, \zeta) \qquad (2.3)$$

によって形式的に定義される．$Bix = \{\bar{1}, 0, 1\}$である．ただし，iとxはBixおよびgixの添字にも対応させられる．すなわち，$Bix = B(i, x)$, $gix = g(i, x)$と考えればよい.外側のノードを加数の高階ストリング・ノード(ノードの外側にノード値をラベル表示），内側のノードを被加数のストリング・ノード（ノードの内側にノード値をラベル表示）と呼ぶ．被加数に関して，有向ブロックgixが定義される．すなわち，有向ブロックgixは，高階ストリング・グラフ$S=(B^2, f, B^3)$によって定義される$p\in B^3$；$f(p)=(\partial r(p), \partial l(p))$；$x=\partial r(p)$, $y=\partial l(p)$に対して，

$$gix(q)=((\partial r(q), \partial l(q)) \mid \partial r(q)\in Bix^2, \partial l(q)\in Bjy^2) \qquad (2.4)$$

によって定義される．ただし，i, j, x, yは添字にも対応させられ，$q\in Bix^3$, Bjy^3；$Bjy = \{\bar{1}, 0, 1\}$である．$j\in\{\bar{1}, 0, 1\}$は1つ上位桁への桁上げ出力状態である．

任意の桁の演算規則は，加数の高階ストリング・ノード$x\in B^2$と被加数のストリング・ノード$y\in Bix^2$に関して定義される．すなわち，現在の桁の結果をrとすると，各ノードの左側の値とキャリー出力状態 j, キャリー入力状態 i によって，

$$r = -2j + \partial l(x) + \partial l(y) + i \ \in\{\bar{1}, 0, 1\} \qquad (2.5)$$

のように表現される．ここでは，ζは現在の桁の結果rを被加数のストリング・ノードyにラベル付ける関数であるとする．図2．4に本除算方式で採用する冗長2進加算規則を表現する高階ストリング・グラフHSを示す.

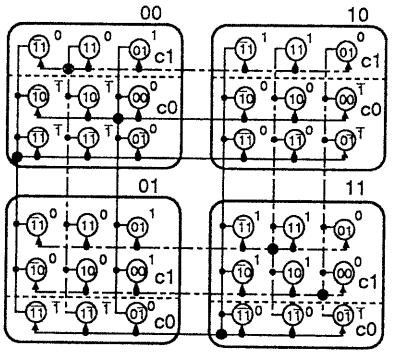

図2．4　本除算方式で採用する冗長2進加算規則を表現する高階ストリング・グラフHS．c1：キャリー1；c0：キャリー0.

3．除算アルゴリズム

3．1節で，まず，除算アルゴリズムについて，3．2節で，商数字を選択する規則を冗長2進加算規則のストリング・グラフ表現を利用して導出する方法について，3．3節で，除数の範囲を正規化区間［1／2，1）より若干ずらせて，商数字選択を簡単にする方法について述べる.

3．1 基数2の除算アルゴリズム

被除数をY，除数をXとする．基数2の除算アルゴリズムは，次のような漸化式で表わされる.

$$R_{i+1} = 2\ (R_i - q_i \cdot X) \qquad (3.1)$$

ここで，iは漸化式のステップ数，$R_0=Y$，R_iはiステップ目の商数字q_iの決定前の部分剰余で，q_iを決定した後に，$R_i-q_i\cdot X$を求めて基数2を乗じてシフトアップしたものがR_{i+1}である．漸化式（3．1）は冗長2進加算器を利用して求めるので，$q_i\in\{\bar{1},\ 0,\ 1\}$である．図2．4に示した冗長2進加算規則を使用する．さらに，数は除数(非冗長2進数)を加減算する計算の都合上，次のような変則的な2の補数表現法をとるものとする．

$$r_{-1}r_0.r_1r_2r_3\cdots=-r_{-1}\cdot 2^1+\sum_{j=0}^{n}r_j\cdot 2^{-j} \qquad (3.2)$$

ここで，$r_{-1}\in\{0,\ 1\}$，その他$r_i\in\{\bar{1},\ 0,\ 1\}$．

3．2　高階ストリング・グラフ表現による商数字選択規則の導出

図2．4の冗長2進加算規則の高階ストリング・グラフ表現を利用して，基数2の除算の商数字q_iの選択規則を導出する方法について説明する．商数字q_iは冗長2進数の1桁$\{\bar{1},\ 0,\ 1\}$で表される．それで，各演算ステップごとに1桁ずつシフトアップするので，シフトアップ前は常に最上位2桁($r_{-1}r_0$桁)がゼロになることを保証しなければならない．そのために，図2．4をもとにして最上位2桁の結果がゼロになるものを最上位4桁分まで展開して図3．1に示すようなものを作る．図3．1の展開では，最上位1桁目（r_{-1}桁）は加数が補数表示にあたるため省略して2桁目（r_0桁）から表示している．また，図2．4の高階ストリング・グラフ表現では，下位桁から上位桁へ向かって有向辺をつけているが，ここでは上位桁から下位桁へたどるため，有向辺を逆に見て展開する．浮動小数点表現では，演算前の数は正規化されていて，0．1…　のかたちをしている．$q_i=1$を選択するときは，加数は11．0…（$=-X$），$q_i=\bar{1}$を選択するときは，加数は00．1…（$=X$），$q_i=0$を選択するときは，加数は00．0…0　または11．1…1　+00．0…01　（$=0$）のかたちに表現されるので，図3．1に示すように，それぞれ（a-1）加数ノード11または（b-1）加数ノードの00，（a-2，3，4）加数ノード10と（b-2，3，4)加数ノード01から出発してストリング・グラフ展開してみる．ここで，部分剰余のシフトアップ前の結果が常に00．…であることを保証しなければならないので，ラベル値が0の被加数ノードから出発するもののみを取り出して表示する．ただし，図ではr_{-1}桁が省略されているが、その桁のラベル値ももちろん0である．また，部分剰余のシフトアップ後の結果が1．1…または$\bar{1}$．$\bar{1}$…となると，q_iの選択が意味的にできなくなるので，これらを除外しなければならない．それで，図3．1において，（a-4）の被加数101…で加数101…の場合に，結果が1．1…に，（b-4）の被加数$\bar{1}0\bar{1}$…で加数010…の場合に，結果が$\bar{1}$．$\bar{1}$…となるが，このような場合は存在しないことが示せる．なぜならば，演算は（a-2）の場合から出発するが，加数が101…のときは結果に1．…または$\bar{1}$．…が出現しないから以降も出現できないからである．

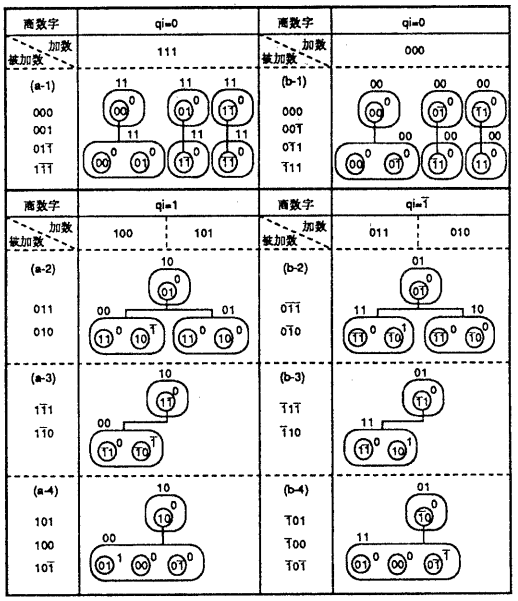

図3．1　冗長2進加算規則の高階ストリング・グラフ表現の上位3桁を展開したもの。ただし、最上位1桁目は補数表示にあたるため省略して2桁目から表示している。

図3．1をまとめると表3．1に示すように部分剰余の上位3桁$r_0r_1r_2$からの商数字q_iの選択と加算制御信号q_cとq_vが得られる．

表３．１　部分剰余の上位３桁$r_0r_1r_2$からの商数字q_iの選択と加算の制御（d：don't care）

入力 $r_0r_1r_2$	商数字 q_i	出力 q_cq_v	入力 $r_0r_1r_2$	商数字 q_i	出力 q_cq_v
$\bar{1}\bar{1}\bar{1}$	d	dd	111	d	dd
$\bar{1}\bar{1}0$	d	dd	110	d	dd
$\bar{1}1\bar{1}$	d	dd	$11\bar{1}$	d	dd
$\bar{1}0\bar{1}$	$\bar{1}$	01	101	1	11
$\bar{1}00$	$\bar{1}$	01	$10\bar{1}$	1	11
$\bar{1}0\bar{1}$	$\bar{1}$	01	$10\bar{1}$	1	11
$\bar{1}1\bar{1}$	$\bar{1}$	01	$1\bar{1}1$	1	11
$\bar{1}\bar{1}0$	$\bar{1}$	01	$1\bar{1}0$	1	11
$01\bar{1}$	$\bar{1}$	01	011	1	11
$0\bar{1}0$	$\bar{1}$	01	010	1	11
$\bar{1}11$	0	00	$1\bar{1}\bar{1}$	0	10
$0\bar{1}1$	0	00	$01\bar{1}$	0	10
$00\bar{1}$	0	00	001	0	10
000	0	d0	000	0	d0

３．３　スケーリング操作による商数字選択の簡単化

M.D. Ercegovac [8,9,11]は，従来のように［１／２，１）の区間に正規化する方法では商数字を選択するために必要な桁数が多くなりすぎて４以上の高基数を使う利点がないことを指摘した．そのため，彼はより少ない桁数で選択する方法を提案した．それは基数４の場合，従来の正規化区間の［１／２，１）より若干ずれた［６３／６４，９／８）の区間に収まるように変換する（彼はこれをスケーリングと呼んだ）ものである．本報告の前節までの高階ストリング・グラフを利用した考察でも，得られた図３．１が非対称であるため、区間をずらせば対称になるのではないかという推測ができる．また，図３．１に示した冗長２進加算規則のストリング・グラフ展開によれば，商数字の選択には上位３桁の部分剰余値を参照しなければならない．しかし，この図から，もし，除数の正規化区間をずらせることができれば，商数字をもっと容易に選択できそうなところがあるかもしれない．以下ではこのことが具体的に示せることを述べる．

図２．４の展開を元にして，商数字$q_i=0$の場合，演算後の部分剰余値の上位１桁r_0がゼロとなるものを表示すると図３．２に示すようになる．そして，商

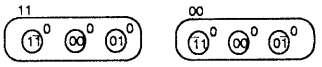

図３．２　商数字$q_i=0$の場合について，部分剰余値の上位１桁r_0がゼロとなる展開．

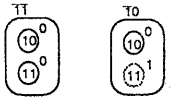

図３．３　商数字$q_i=\bar{1}$の場合から除数のスケーリング範囲の候補を求める：点線で示した加数ノードは候補から除外される．

数字$q_i=\bar{1}$の場合について，演算前の部分剰余値の残りのパターンは$0\bar{1}.\ \bar{1}\cdots$，$0\bar{1}.\ 0\cdots$であるから，演算後の部分剰余値の上位１桁r_0桁がゼロとなる加数のパターンを，図２．４の加数と被加数の関係を逆にして展開して求めると，図３．３に示すように，加数が$01.\ 0\cdots$の場合しかないことがわかる．従って，除数の範囲は$1.\ 0\cdots=[1,\ 3/2)$しかあり得ないことがわかる．しかも部分剰余値の上位２桁を参照するだけで商数字が選択できる解が得られる．最上位桁の補数表現部分は必要ない．この結果をまとめると図３．４に示すようになり，さらに表３．２に示すように部分剰余の上位２桁r_0r_1のみからの商数字q_iの選択と加算制御信号q_cとq_vが得られる．そして，スケーリング操作は，

$$M = m_1 2^1 + m_0 2^0 + m_{-1} 2^{-1} \qquad (3.3)$$

とするとき，

$$Y/X = (MY)/(MX) = Y'/X' \qquad (3.4)$$

なる関係から，表３．３に示すようにもとの除数Xの小数点以下２桁目を反転＋１させる操作によって求め

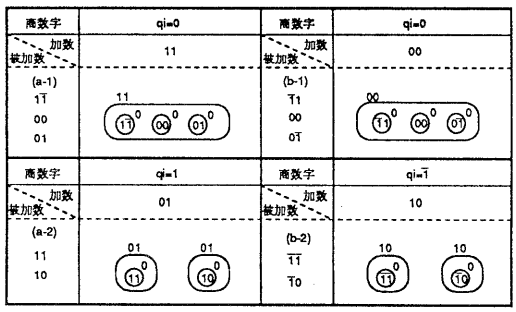

図３．４　冗長２進加算規則の高階ストリング・グラフ表現の上位２桁r_0r_1を展開したもの。

ることができる．ただし，この場合は実際にはもっと簡単で，除数を正規化したときに，小数点以下２桁目の値を参照するだけで決定できる．ここで，Y'=MY，X'=MX.

表３．２　部分剰余の上位２桁r_0r_1からの商数字q_iの選択と加算の制御

入力 r_0r_1	商数字 q_i	出力 q_cq_v	入力 r_0r_1	商数字 q_i	出力 q_cq_v
$\overline{1}\overline{1}$	$\overline{1}$	01	11	1	11
$\overline{1}0$	$\overline{1}$	01	10	1	11
$\overline{1}\overline{1}$	0	00	$1\overline{1}$	0	10
$0\overline{1}$	0	00	01	0	10
00	0	d0	00	0	d0

表３．３　除数Ｘのスケーリング操作

除数Ｘ	$2^1 2^0 2^{-1}$
0.11	0 1 1
0.10	1 0 0

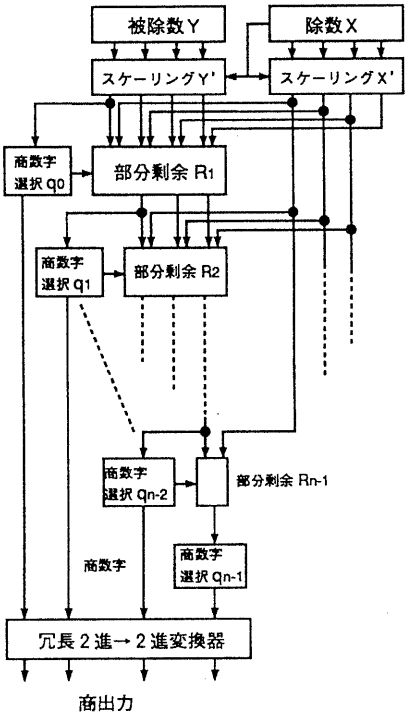

図４．１　除算器の構成

４．基数２の除算器の実現

基数２の除算器は，図４．１に示すように各演算ステップの回路を配列的に接続して構成される．まず，被除数Ｙおよび除数Ｘは，表３．３に示した除数Ｘのスケーリング操作Ｍにもとづいて，図４．２に示すように除数Ｘの小数点以下２桁目の値x_2によって入力がセレクトされ，桁上げ先見付き加算器（ＣＬＡ）を使って，それぞれＹ'，Ｘ'に変換される．一般に，ゼロ除算の検出や除数Ｘと被除数Ｙの桁合わせの操作が必要なので，これらと並列にスケーリング操作を実行すればオーバヘッドの問題はない．

部分剰余の上位２桁r_0r_1から商数字q_iが選択される．この商数字選択回路および加算制御回路は表３．２から図４．３に示すものが得られる．比較のため，表３．１から得られる従来の３桁解の商数字選択回路を図４．４に示す．この結果，商数字選択のためのゲート段数（２入力ＡＮＤ，ＯＲ，ＥＯＲ相当を１段と数える）は，従来の４段から２段に削減される．そして，商数字１桁あたりの演算に必要な総ゲート段数は，９段から７段になり，約２５％改善されることになる．

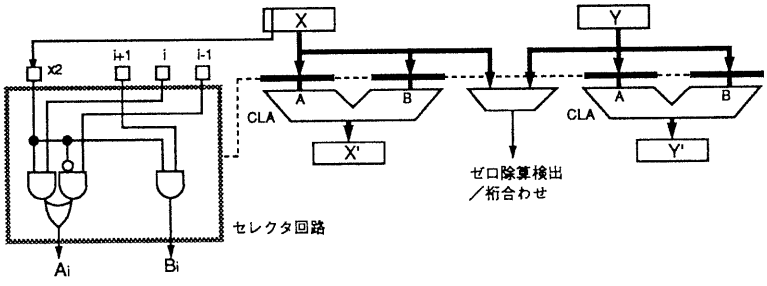

図４．２　オペランドのスケーリング操作

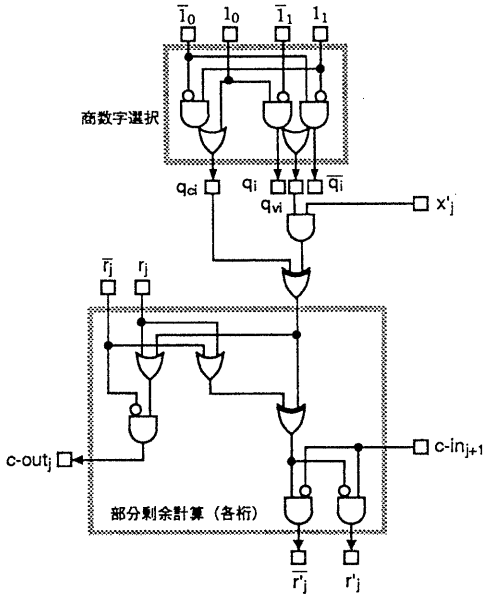

図４．３　商数字選択（２桁）と部分剰余計算回路

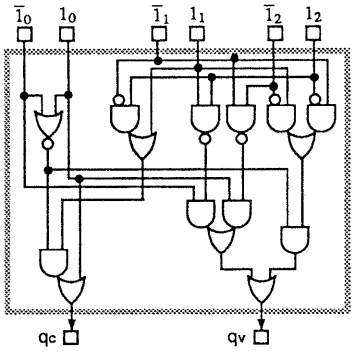

図４．４　商数字選択回路（３桁）

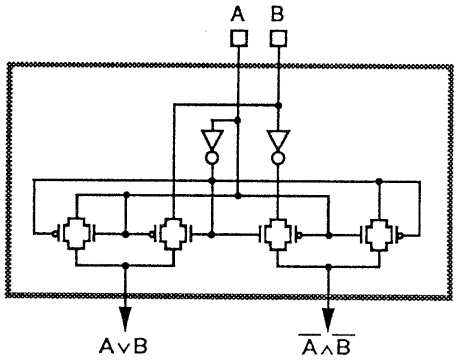

図４．６　パス・トランジスタを使った基本論理回路

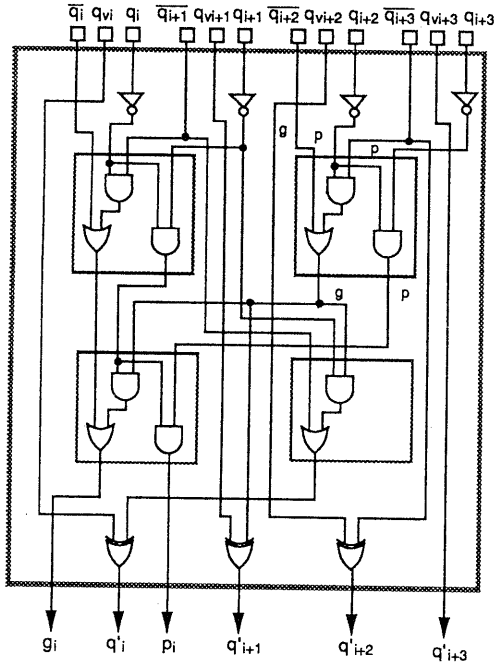

図４．５　冗長２進→２進変換器（４桁分）

商数字q_iの選択結果は，冗長２進数であるため，通常の２進数に変換される．商数字は逐次得られることから，M.D. Ercegovac [7]は逐次演算中に変換する方法を考案している．彼の方法は，２つの変換候補を計算しながら商数字の決定に従って候補を選択する方法である．本報告では，冗長２進数は1と$\bar{1}$が互いに排他的に表現されるという性質を利用して，図４．５（４桁分の例）に示すように，論理を簡単にした冗長２進→２進変換器を構成する．また，図４．６に示すパス・トランジスタを使った基本論理回路によってこれらの論理ゲートを構成すれば、ゲートのスイッチングの役割をしているＡ入力側（商数字の上位桁に相当）の動作が先に完了しているとすれば，Ｂ入力側（商数字の下位桁に相当）の動作がその後に定まっても，Ｂの入力信号は瞬時にゲートを通過するだけなので高速に動作することになる．

5．おわりに

基数２の除算器の構成に関して，従来のように除数を［１／２，１）の区間に正規化する方法では，商数字を選択するためには部分剰余値の３桁以上を参照する必要がある．ところが，外村の提案している高階ストリング・グラフ手法を応用することによって，除数の範囲を，［１，３／２）の区間にスケーリング操作すると，部分剰余値の２桁を参照するだけで商数字が選択できることがわかった．その結果，従来よりも総ゲート段数が約２５％減少し効率的な除算器が構成できることを示した．この解は，最近，N. Burgess [4]が1963年に発表されたSvoboda のアルゴリズム[16,23]を適用して得ているが，それを再確認するものである．ただし,彼の方法は商数字に０を選択したとき,ノー・オペレーションで桁あふれを防ぐのに対して，本方法は，この場合もゼロ加算を実行し，桁あふれのないことを保証するものである．ノー・オペレーションのためのセレクタは不要である．

本報告で示した基数２の除算器の理論および構成方法は，今後，基数が２より高いものに拡張されるが，それらを議論するときの基礎となるものである．そして，それらの評価基準となる．

参考文献

[１] Atkins, D.E. : Higher-Radix Division Using Estimates of the Divisor and Partial Remainders, IEEE Trans. on Computers, Vol. C-17, No.10, pp.925-934, Oct. (1968).

[２] Atkins, D.E. : Design of the Arithmetic Units of ILLIAC III : Use of Redundancy and Higher Radix Methods, IEEE Trans. on Computers, Vol.C-19,No.8, pp.720-733, Aug. (1970).

[３] Beame, P.W., Cook, S.A. and Hoover H.J. : Log Depth Circuits for Division and Related Problems,25th IEEE FOCS, pp.1-6,Oct. (1984), SIAM J. Comput., Vol.15, No.4, pp.994-1003, Nov. (1986).

[４] Burgess, N. : A Fast Division Algorithm for VLSI, ICCD '91, pp.560-563, Oct. (1991).

[５] Edamatsu, H. et al. : A 33MFlops Floting Point Processor using Redundant Binary Representation, Proc. of ISSCC88, pp. 152-153, (1988).

[６] Ercegovac, M.D. and Lang, T. : A Division Algorithm with Prediction of Quotient Digits, IEEE Proc. of 7th Symposium on Computer Arithmetic, pp.51-56, (1985).

[７] Ercegovac, M.D. and Lang, T. : On-the-Fly Conversion of Redundant into Conventional Representations, IEEE Trans. on Computers, Vol.C-36,No.7, pp.895-897, July (1987).

[８] Ercegovac, M.D. and Lang, T. : Simple Radix-4 Division with Divisor Scaling, UCLA Comput. Sci. Dept., Tech. Rep. CSD-870015, Mar. (1987).

[９] Ercegovac, M.D. et al. : Implementation of Fast Radix-4 Division with Operands Scaling, ICCD '88, 486-489, (1988).

[10] Ercegovac, M.D. and Lang, T. : Fast Radix-2 Division with Quotient-Digit Prediction, Journal of VLSI Signal Processing, 1, pp.169-180, (1989).

[11] Ercegovac, M.D. and Lang, T. : Simple Radix-4 Division with Operands Scaling, IEEE Trans. on Computers, Vol.C-39, No.9, pp.1204～1208, Sept. (1990).

[12] Hwang, K., Computer Arithmetic, John Wiley & Sons, Inc., New York, 1979. (堀越彌監訳：コンピュータの高速演算方式，近代科学社，東京，1980.)

[13] Kuninobu, S. et al. : Design of High Speed MOS Multiplier and Divider using Redundant Binary Representation, IEEE International Symposium on Computer Arithmetic, pp80-86, (1987).

[14] 岡部寿男，高木直史，矢島脩三 ： 剰余数表示法を用いたO（log n）段ｎビット２進除算器， 信学技報，ＡＬ８５－８９，pp.35-44, Mar. (1986).

[15] 岡部寿男，高木直史，矢島脩三 ： 剰余数表示法を用いた初等関数計算の対数段回路アルゴリズム，信学論（Ｄ），J73-D-I, No.9, pp.723-728, (1990).

[16] A. Svoboda : An algorithm for division, Information Processing Machines, Vol. 9, pp. 183-190, 1963.

[17] 高木直史 他２名：冗長２進表現を利用したＶＬＳＩ向き高速除算器，信学論（Ｄ），J67-D, No.4, pp.450-457, (1984).

[18] 高木直史，矢島脩三 ： 算術演算のハードウェアアルゴリズム， 情報処理 Vol.26, No.6, pp.632-639, (1985).

[19] 高木直史 他２名：演算処理装置，公開特許公報(Ａ) 昭63-8824, 昭63-49836, (1988).

[20] 谷口隆志，枝末壽一，西山保，國信茂朗：冗長２進表現を用いた高速浮動小数点プロセッサ， 信学技報, ICD87-100, pp.43-48, (1988).

[21] Taylor, G.S. : Radix 16 SRT Dividers With Overlapped Quotient Selection Stages, IEEE Proc. of 7th Symposium on Computer Arithmetic, pp.64-71, (1985).

[22] 外村元伸 ： 演算アルゴリズムのストリング・グラフ表現，情報処理学会論文誌，Vol.31, No.8, pp.1242-1250, (1990).

[23] Tung, C. :A division algorithm for signed-digit arithmetic, IEEE Trans. Computers, Vol. 17, pp. 887-889, (1968).

[24] 矢島脩三 ： ハードウェアアルゴリズムの最近の動向，電子情報通信学会誌 Vol.72, No.8, pp.907-914, (1989).

[25] Zurawski, J.H.P. and Gosling, J.B. : Design of High-Speed Square Root Multiply and Divide Unit, IEEE Trans. on Computers, Vol.C-36,No.1, pp.13-23, Jan. (1987).