演算アルゴリズムのストリング・グラフ表現†

外村 元伸††

コンピュータ・グラフィックス分野が急激に発展してきたので，32ないし64ビットの浮動小数点演算がごく普通に要求されるようになってきた．このように処理桁数が増加してくると，桁上げ伝播による遅延が問題になり，桁上げ伝播のない演算方式が注目されている．従来，桁上げ保存加算器が主として桁上げ伝播のない演算方式に利用されていた．しかし，最近，$\{-1, 0, +1\}$ で表現する冗長2進加算器を利用する方が，相性がよく，すぐれた規則性をもち，加減乗除，開平などの高速演算アルゴリズムを導出できることがわかってきた．本論文では，これらの演算アルゴリズムをストリング・グラフと呼ぶ表現によって視覚化または形式化することを提案する．そして，このストリング・グラフ表現を使って，演算アルゴリズムの解析や証明が容易にできることを示す．まず，Boothのアルゴリズムをストリング・グラフ表現することから試みる．次に，正準リコード・アルゴリズムのストリング・グラフ表現を得るために，グラフを高階化する方法を示す．そして，この高階グラフ表現を用いて冗長2進加算アルゴリズムのストリング・グラフ表現を導出する．さらに，冗長2進加算アルゴリズムのストリング・グラフ表現が，除算アルゴリズムを導出するときの証明に利用できることを示す．

1. はじめに

コンピュータ・グラフィックス分野が急速に成長し，しかも，32ビット（単精度）ないし64ビット（倍精度）の浮動小数点演算がごく普通に要求されるようになってきた．このように処理桁数が増加してくると，すべての算術演算の基本となるのは加算であるため，加算の桁上伝播による遅延が本質的に高速化を妨げるようになってくる．ここで問題となる遅延は処理桁数nに依存し，一定の遅延は本質的とは考えられないので，以降，遅延には処理桁数nに依存するもののみを主として考えることにする．桁上げを一時的に保存しておいて，それを次の加算時に加える桁上げ保存加算器が遅延をなくす方法としてよく知られている．また，桁上げ伝播が高々1回で加算ができる符号付き2進数$\{-1, 0, +1\}$を使う方法が古くAvizienis[2]によって考案されている．符号付き2進数表現は命令の内部処理に限定して利用される場合が主として多い．その場合，冗長な－1を追加するという意味で，今日，冗長2進数表現と呼ばれているので，以降はこれに統一することにする．冗長2進数表現を使う方法に関して，最近，高木直史ら[3)~5),18)]，原田義久ら[6),7)]，仲西正ら[12),13)]，谷口隆志・枝末壽一・西山保・國信茂朗[14)~17)]の一連の研究で著しい進展があり，乗除算器その他に応用された．

本論文では，加減乗除など種々の算術演算アルゴリズムをストリング・グラフなるもので表現することを提案して，より効率的な演算アルゴリズムを見出すための解析や証明に応用する基礎付けを行う．そして，上記のように最近注目されている冗長2進数体系に主として注目し，ストリング・グラフ表現によって解析や証明を行い，現在までに得られている上記結果を含めて理論的な整理を行う．

2. 準備と定義

2.1 予備知識

数の記述表現を $y=\sum_{i=1}^{n-1} x_i \cdot 2^i$ とすると，同じ数を表すにも，$x_i \in \{0, 1\}$ や $x_i \in \{-1, 0, 1\}$ などと選ぶことによって，その表現体系が異なってくる．$x_i \in \{0, 1\}$ にすれば通常の2進数表現であり，$x_i \in \{-1, 0, 1\}$ にすれば冗長2進数表現である．これらの数表現体系を区別するために，$\bar{1}=-1$ と記号定義して，

$x_i \in \{0, 1\}$ のとき，$[x_{n-1}x_{n-2}\cdots x_1x_0]_2$

$x_i \in \{\bar{1}, 0, 1\}$ のとき，$[x_{n-1}, x_{n-2}\cdots x_1x_0]_{RB2}$

と表記することにする．ここで，RBは冗長2進数（Redundant-Binary）の略である．

明らかに，$x_i \in \{0, 1\}$ ならば，

$$[x_{n-1}x_{n-2}\cdots x_1x_0]_2=[x_{n-1}x_{n-2}\cdots x_1x_0]_{RB2}$$

が成り立つ．

また，通常の2進数を2の補数表現で扱う場合は，$[x_{n-1}x_{n-2}\cdots x_1x_0]_2$ の値をyとすると，

$$y=-x_{n-1}\cdot 2^{n-1}+\sum_{i=0}^{n-2} x_i \cdot 2^i \qquad (2.1)$$

† String Graph Representations for Arithmetic Algorithms by MOTONOBU TONOMURA (Hitachi Central Research Laboratory, Hitachi Ltd.).

†† (株)日立製作所中央研究所

であるから，$x_i \in \{0, 1\}$ ならば，

$$[x_{n-1}x_{n-2}\cdots x_1x_0]_2 = [\widetilde{x_{n-1}}x_{n-2}\cdots x_1x_0]_{RB2}$$

が成り立つ．ただし，$\widetilde{x_{n-1}}$ は，$x_{n-1}=0$ のとき 0，$x_{n-1}=1$ のとき $\bar{1}$ を表す．

2.2 2次の Booth アルゴリズムと正準リコード・アルゴリズム

冗長2進数表現を利用した数の表現例として，まず乗算の部分積を求めるために使われる2次の Booth アルゴリズムに関して説明する．式(2.1)において，$x_i \in \{0, 1\}$ として，次のような式変形によって導ける[8]．ただし，便宜のため n は偶数であるとし，また $x_{-1}=0$ とする．

$$\begin{aligned} y &= -x_{n-1}\cdot 2^{n-1} + \sum_{i=0}^{n/2-1} x_{2i}\cdot 2^{2i} + \sum_{i=0}^{n/2-1} x_{2i-1}\cdot 2^{2i-1} \\ &= -x_{n-1}\cdot 2^{n-1} \sum_{i=0}^{n/2-1} x_{2i}\cdot 2^{2i} + \sum_{i=0}^{n/2-1} x_{2i-1}\cdot 2^{2i} \\ &\quad - \sum_{i=1}^{n/2-1} x_{2i-1}\cdot 2^{2i-1} \end{aligned}$$

ここで，第1項と第4項をいっしょにして，

$$\begin{aligned} y &= -2\sum_{i=0}^{n/2-1} x_{2i+1}\cdot 2^{2i} + \sum_{i=0}^{n/2-1} x_{2i}\cdot 2^{2i} \\ &\quad + \sum_{i=0}^{n/2-1} x_{2i-1}\cdot 2^{2i} \\ &= \sum_{i=0}^{n/2-1} (-2x_{2i+1} + x_{2i} + x_{2i-1})\cdot 2^{2i} \end{aligned} \qquad (2.2)$$

このように，式(2.2)により，冗長2進数表現を使って2次の Booth アルゴリズムが表現される．すなわち，

$$[z_{2i+1}z_{2i}]_{RB2} = -2[x_{2i+1}]_2 + [x_{2i}]_2 + [x_{2i-1}]_2 \qquad (2.3)$$

ここで，$z_{2i+1}z_{2i} \in \{00, 01, 10, 0\bar{1}, \bar{1}0\}$ であることは容易にわかる．

ところで，冗長2進数による表現は一意でないため，式(2.3)の Booth アルゴリズムの表現のほかにも様々な性質をもった表現法が知られている．その中で，非ゼロの数が減少する性質をもつものを考える．$k, i \geqq 0$ に対して，

$$2^{k+i+1} - 2^k = 2^{k+i} + 2^{k+i-1} + \cdots + 2^k \qquad (2.4)$$

なる基本的な変換によって連続する1の数を減少させることができる．正準リコード・アルゴリズム[1]と呼ばれるものは，この原理を利用しており，乗数を下位の桁から順に見ていって，1が2つ連続したら式(2.4)のように桁上げし，現在の桁を $\bar{1}$ に変換する．以下，この桁上げ分も加えて，同様の操作を上位桁に向かって繰り返す．正準リコード・アルゴリズムの性質の導出については，後に述べる．

2.3 桁上げ伝播なし加算の原理

冗長2進数表現を利用したもう1つの例として，被加数 $[x_{n-1}x_{n-2}\cdots x_1x_0]_2$ と加数 $[y_{n-1}y_{n-2}\cdots y_1y_0]_2$ の加算について考える．各桁ごとに同時に加算を行うとし，x_i と y_i の桁に注目する．下位からの桁上げが上位に伝播する場合は，x_i または y_i のどちらか一方のみが1のときに，1つ下位の桁から桁上げが生じたときに起こる．この場合，冗長2進数表現を使って，

$$x_i + y_i = 0 + 1 = [1\ \bar{1}]_{RB2}$$

と考えてあらかじめ1つ上位に桁上げを行っておけば，1つ下位の桁 x_{i-1}, y_{i-1} からの桁上げは，$\bar{1}$ によって x_i, y_i の桁位置で吸収することができる．したがって，各桁位置での加算は，少なくとも一方が1のときに1つ上位の桁へ桁上げを行うことにしておき，一時的に冗長2進数表現をしておけば，2つ以上上位の桁への桁上げ伝播は生じないことになる．実際には，次の計算入力において，$\bar{1}$ が現れるので，$\bar{1}$ の入力も含めた演算規則の場合については，後に述べることにする．また，冗長2進数表現で得られた結果 $[z_{n-1}z_{n-2}\cdots z_1z_0]_{RB2}$ を通常の2進数 $[u_{n-1}u_{n-2}\cdots u_1u_0]_2$ に変換する必要がある場合には，値 $v_i \in \{0, 1\}$ をもつ1表現と呼ぶ非負部分 $[v_{n-1}v_{n-2}\cdots v_1v_0]_{RB2}$ と，値 $w_i \in \{\bar{1}, 0\}$ をもつ $\bar{1}$ 表現と呼ぶ非正部分 $[w_{n-1}w_{n-2}\cdots w_1w_0]_{RB2}$ に分離し，通常の加算器，例えば，桁上げ先見加算器を用いて減算すればよい．すなわち，

$$\begin{aligned} &[z_{n-1}z_{n-2}\cdots z_1z_0]_{RB2} \\ &= [v_{n-1}v_{n-2}\cdots v_1v_0]_{RB2} + [w_{n-1}w_{n-2}\cdots w_1w_0]_{RB2} \\ &= [v_{n-1}v_{n-2}\cdots v_1v_0]_2 \\ &\quad - [-[w_{n-1}w_{n-2}\cdots w_1w_0]_{RB2}]_2 \\ &= [u_{n-1}u_{n-2}\cdots u_1u_0]_2 \end{aligned} \qquad (2.5)$$

2.4 グラフ表現の数学的な準備とその定義

この節では，前述の演算アルゴリズムなどをグラフ論的手法で記述するための数学的な準備と定義を与える（主として，文献9）に従った）．まず，Xが集合のとき，$P(X) = \{x \mid x \subseteq X\}$ は X のベキ集合と呼ばれ，Xの部分集合すべての集りを表す．また，$|X|$ は集合 X の濃度を表す．集合 $X_1, \cdots, X_m$ の直積は，$X_1 \times \cdots \times X_m = \{(x_1, \cdots, x_m) \mid i = 1, \cdots, m$ に対して，$x_i \in X_i\}$ によって定義される．そして，X から Y への関数 f は，すべての $x \in X$ に対して，$|f \cap (\{x\} \times Y)| = 1$ となるような $X \times Y$ の部分集合によって定義される．これを，$f: X \to Y$ で表す．そして，各 $x \in X$ に対

して，$f(x)$ は $f\cap(\{x\}\times Y)$ の第2成分の一意な要素である．すべての $y\in Y$ に対して，$|f\cap(X\times\{y\})|\leqq 1$ ならば単射，$|f\cap(X\times\{y\})|\geqq 1$ ならば全射，そして $|f\cap(X\times\{y\})|=1$ ならば双射という．普通，例えば，単射は $x_1, x_2\in X$ に対して，$f(x_1)=f(x_2)$ ならば $x_1=x_2$ と定義されるがここでの定義と同値である．

一般にグラフ G とは3項組 (V,f,E) のことである．ここで，V と E は互いに素な集合であり，V の要素はノード（頂点），E の要素はブロック（辺）と呼ばれる．そして，f は単射であり，$f: E\to P(V)$ である．$x\in E$ に対して，$|f(x)|$ はブロック x のサイズという．そして，すべての $x\in E$ に対して，ブロック x のサイズが k のとき，グラフ G はブロックサイズ k をもつという．普通は，ブロックサイズ2のものを単にグラフと呼んでいる．V と E が集合であり，かつ $f: E\to P(V)$ ならば，$f^*(y)=\{x\in E\mid y\in f(x)\}$ によって，f の転置と呼ばれる関数 $f^*: V\to P(E)$ が定義される．任意の $y\in V$, $x\in E$ に対して，$y\in f(x)\Leftrightarrow x\in f^*(y)$ なので，$f^{**}=f$ である．そして，グラフ $G=(V,f,E)$ に対して，グラフ $G^*=(E,f^*,V)$ は G の転置と呼ばれる．また，グラフ $G=(V,f,E)$ に関して，

1) 任意の $x\in E$ に対して，$f(x)\neq\phi$,

2) 任意の $y\in V$ に対して，$f^*(y)\neq\phi$

なる2つの性質を付加したものは，Berge 流[10]にいえば高階グラフと呼ばれる．なぜならば，上の定義 2) に相当する Berge の高階グラフの定義 2′) は，

2′) $\bigcup_{x\in E} f(x)=V$

であるが，ある $x\in E$ に対して，

$$y\in V\Leftrightarrow y\in f(x)$$
$$\Leftrightarrow x\in f^*(y)\Leftrightarrow f^*(y)\neq 0$$

であるから，ここでの定義と同値である．

3. 演算アルゴリズムのストリング・グラフ表現

3.1 ストリング・グラフ表現

A をアルファベットと呼ばれる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．そして，A 上の空も含むすべてのストリングの集合は，A^* によって表される．非ゼロ整数 n に対して，A^n は A 上の長さ n のストリングすべての集合を表す．$a, b\in A^*$ 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 $a\in A$ と $x\in A^*$ に対して，演算子 ∂_r と ∂_l を

$$\partial_r(ax)=\partial_l(xa)=x \tag{3.1}$$

によって定義する．そして，ストリング・グラフ[23]と呼ぶ4項組 $S=(A^{n-1}, f, A^n, \zeta)$ が定義される．ここで，$n>0$ で，A^{n-1} がノードの集合で，A^n がブロックの集合である．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・ブロックと呼ぶ．f は任意の $x\in A^n$ に対して，ブロックサイズ2の

$$f(x)=(\partial_r(x), \partial_l(x)) \tag{3.2}$$

によって与えられる．そして，$f(x)$ に $\partial_r(x)$ から $\partial_l(x)$ 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 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．このストリング・グラフ表現は，M. Nasu[11] がセルラ・オートマトンの解析に使用したものにもとづいている．古くは De Bruijn[19] の論文で扱われたので De Bruijn グラフとも呼ばれている．ストリング・グラフ表現はシフト・レジスタ[20]，ネットワーク[21),22)]の解析などにも使われている．

3.2 演算アルゴリズムのストリング・グラフ表現

ストリング・グラフ表現の具体的な例として，2.2節で述べた2次の Booth アルゴリズムを**図1**に示すようなストリング・グラフ

$$\mathrm{BOOTH}=(B^2, f, B^3, \zeta) \tag{3.3}$$

で表現する．ここで，$B=\{0,1\}$ である．式(2.3)から，有向ブロックは，ストリング・ブロック $x_{2i+1}x_{2i}x_{2i-1}$ に対して，$f(x_{2i+1}x_{2i}x_{2i-1})=(x_{2i}x_{2i-1}, x_{2i+1}x_{2i})$ と表現できるので，ζ は有向ブロック $f(x_{2i+1}x_{2i}x_{2i-1})$ に $[x_{2i+1}x_{2i}]_{RB2}$ とラベル付ける関数であると定義できる．図1の有向ブロックのラベル値は非ゼロが1桁であることを表現している．この性質は，部分積を求めるときに，乗数または被乗数の単純なシフト動作とその結果の加減算で実現できるために重要である．

次に，Booth アルゴリズムとは異なる冗長2進数表

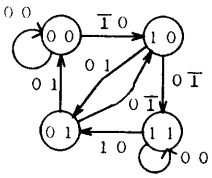


図1 2次の Booth アルゴリズムのストリング・グラフ表現
$\mathrm{BOOTH}=(B^3, f, B^3, 1)$

Fig. 1 String graph representation for second-order Booth algorithm.

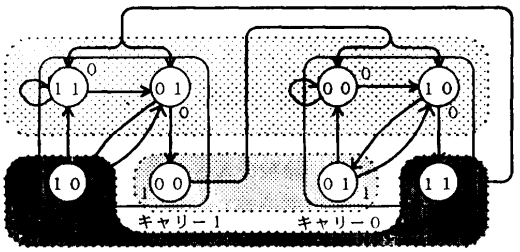


図 2 正準リコードのストリング・グラフ表現
CANON=($\{B_i^2\}$, $\{f_i\}$, B^3, ζ)
Fig. 2 String graph representation for canonical recode.

現をもち，同じく乗算の部分積を求めるために使われる正準リコード・アルゴリズムに関して，ストリング・グラフ表現化することを試みる．正準リコード・アルゴリズムでは，式(2.4)で表されるような桁上げを扱わなければならない．これをグラフ表現するために，桁上げが生じた（キャリー 1）か，生じなかった（キャリー 0）かによって別々のノードの集まりをそれぞれ定義する．すなわち，式(3.3)のストリング・グラフ BOOTH を拡張して，**図 2** に示すように，$i=0,1$ に対して，

$$\mathrm{CANON}=(\{B_i^2\},\{f_i\},B^3,\zeta) \tag{3.4}$$

によって表現する．ここで，キャリー i のノードの集合を B_i^2 とする．ただし，B_i, $B=\{0,1\}$ とする．そして，B_i^2 を高階ストリング・ノードと呼ぶ．有向ブロックは，$x\in B^3$ に対して，

$$f_i(x)=((\partial_r(x),\partial_l(x))\,|\,\partial_r(x)\in B_i^2,\partial_l(x)\in B_j^2) \tag{3.5}$$

によって定義される．ここで，ストリング・ノード $y=\partial_r(x)$ に対して，$\partial_r(y)$ が現在の桁を，$\partial_l(y)$ が1つ上の桁を表す．そして，B_j^2 は正準リコード・アルゴリズムに従って，

$\partial_l(x)\in B_0^2$
…桁上げが生じなかった（キャリー 0）とき，
$\partial_l(x)\in B_1^2$
…桁上げが生じた（キャリー 1）とき
(3.6)

のように選択される．また，ζ はストリング・ノード $y=\partial_r(x)$ に現在の桁 $\partial_r(y)$ の冗長 2 進表現値をラベル付ける関数である．図2に示すように，ノードのラベル値に従ってノードの集まりを区別すると，非ゼロ値がラベル付けされたノードからは，非ゼロ値がラベル付けされたノードへの直接パスが存在しないことから，非ゼロ値が連続することがないという性質が導ける．

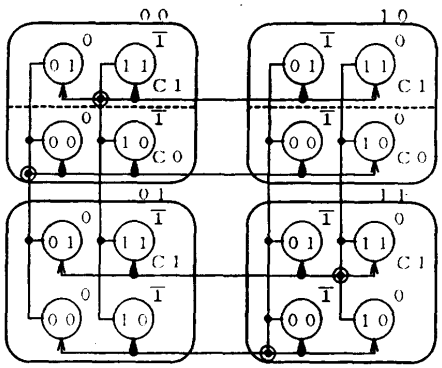


図 3 冗長2進数表現による加算の高階ストリング・グラフ表現例：内側のノードが加数を，外側のノードが被加数を表す．
RBG=($\{B_i^2\}$, $\{f_i\}$, B^3, ζ), $C1$ (キャリー 1), $C0$ (キャリー 0).
Fig. 3 Example of hyper string graph representation for addition by redundant binary representation.

冗長2進数表現を利用したさらにもう1つの例として，2.3 節で被加数 $[x_{n-1}x_{n-2}\cdots x_1x_0]_2$ と加数 $[y_{n-1}y_{n-2}\cdots y_1y_0]_2$ の加算アルゴリズムについて述べたので，ここではこの冗長2進加算アルゴリズムの高階ストリング・グラフ表現について述べる．すなわち，前述のストリング・グラフ CANON をさらに拡張して，**図 3** に示すように，$i=0,1$（キャリー i の入力状態に対応する）に対して，

$$\mathrm{RBG}=(\{B_{ix}^2\},\{f_{ix}\},B^3,\zeta) \tag{3.7}$$

によって表現する．ただし，B_{ix}, $B=\{0,1\}$ である．ここで，まず，被加数の桁移動の流れをストリング・グラフ表現するために高階ストリング・グラフ (B^2, g, B^3) を導入する．$p\in B^3$ に対して，$g(p)=(\partial_r(p),\partial_l(p))$ である．$x\in B^2$ は B_{ix} の添字 x に対応づけられるものとする．B^2 の要素 x，あるいは B_{ix}^2 は被加数の高階ストリング・ノードと呼ぶ．B_{ix}^2 の要素は加数のストリング・ノードである．図3では，外側のノードが被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）を，内側のノードが加数のストリング・ノード（ノードの内側にノード値をラベル表示）を表している．有向ブロック f_{ix} は，高階ストリング・グラフ (B^2, g, B^3) を定義する $p\in B^3$: $g(p)=(\partial_r(p),\partial_l(p))$; $x=\partial_r(p)$, $y=\partial_l(p)$ に対して，

$$f_{ix}(q)=((\partial_r(q),\partial_l(q))\,|\,\partial_r(q)\in B_{ix}^2,\partial_l(q)\in B_{jy}^2) \tag{3.8}$$

によって定義される．ただし，$q\in B_{ix}^3$, B_{jy}^3 である．j は1つ上位桁へのキャリー出力状態である．まず，

このようにして定義された高階ストリング・グラフ RBG=($\{B_{i,x}{}^2\}$, $\{f_{i,x}\}$, B^3, ζ) によって誘導される冗長2進数表現による演算規則について説明する．演算規則は，被加数の高階ストリング・ノード $x\in B^2$ と加数のストリング・ノード $y\in B_{i,x}{}^2$ を使って，キャリー入力状態 i とキャリー出力状態 j を決定することによって定義される．まず，キャリー出力状態 j を決定する桁上げ規則は，現在の桁の結果 s が冗長2進数表現値の許容範囲内であれば選択できる．すなわち，

$$s=-2j+\partial_1(x)+\partial_1(y)+i\in\{\bar{1},0,1\} \tag{3.9}$$

である．ここで，各ストリング・ノードに対して，現在の桁が ∂_1，その1つ下位が ∂_r によって表されるものとする．キャリー入力状態 i を加える前の現在の桁の一時的な和 m は中間和と呼ばれるが，ζ は加数のストリング・ノード y に現在の桁の中間和 m をラベル付ける関数であるとする．中間和 m とキャリー入力状態 i を加えることによって現在の桁の結果 s を求めることができるが，式(3.9)はこの加算によって桁上げ入力が吸収できるもの，すなわち上位桁への桁上げ伝播が生じない演算規則を選択することを意味している．そして，これらは一般に高階ストリング・グラフ RBG によって解析し，決定していくことができる．図3の例は単純で，一般に入力と出力の両方の有向ブロック f_i がともに存在しない加数ノードについては式(3.9)を満たすことができないので表示しないことにすると，

$$\begin{aligned} j&=\partial_1(q)\vee\partial_1(y),\\ m&=-(\partial_1(q)\oplus\partial_1(y)),\\ s&=m+i,\quad i=0,1 \end{aligned} \tag{3.10}$$

なる演算規則が導かれる．ここで，$\vee$ は論理和演算を，$\oplus$ は排他的論理和演算を示す．ところで，図3の例では，最初の入力には $\bar{1}$ 表現が現れないが，出力結果には $\bar{1}$ 表現が現れ，次の演算段階の入力には $\bar{1}$ 表現が現れるため，閉じた演算体系になっていない．そこで，次章では，$B_{i,x}=\{\bar{1},0,1\}$，$B=\{0,1\}$ とし，加算の一方のみに $\bar{1}$ 表現の入力を許す閉じた演算体系について考える．これは，特に除算，開平アルゴリズムにおいて使用されるので，次章では除算アルゴリズムの導出においてストリング・グラフ表現法を応用することについて述べる．

4. 除算アルゴリズムへの応用

4.1節で，まず，除算アルゴリズムの概要について述べる．そして，4.2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証明しながら導出する方法について述べる．

4.1 除算アルゴリズムの概要

一般性を損なわない程度に，符号なし2進小数を仮定する．被除数を Y，除数を X とする．そして，商を Q とし，i 桁目までの部分商を

$$Q_i=\sum_{j=0}^{i}q_j2^{-j} \tag{4.1}$$

とする．また，

$$\begin{aligned} D_{i+1}&=Y-Q_i\cdot X\\ &=D_i-q_i2^{-i}\cdot X \end{aligned} \tag{4.2}$$

なる部分剰余 D_i を導入する[24]．部分剰余列 $\{D_i\}$ はステップ i が増加するにつれて，ゼロに近づき，列 $\{Q_i\}$ は商 Q に近づく．ここで，筆算における除算の手順とは逆に，除数を固定し，各ステップごとに D_{i+1} を基数2でスケーリングしながらシフト・アップしていく．すなわち，$i+1$ ステップ目にシフト・アップ・スケーリングされた部分剰余 R_{i+1} は

$$R_{i+1}=D_{i+1}\cdot 2^{i+1} \tag{4.3}$$

で表される．したがって，除算のアルゴリズムは，次のような漸化式で表せることがわかる．

$$\begin{aligned} R_{i+1}&=(D_i-q_i2^{-i}\cdot X)\cdot 2^{i+1}\\ &=2\cdot(R_i-q_i\cdot X) \end{aligned} \tag{4.4}$$

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，R_{i+1} はそれを決定した後に，$R_i-q_i\cdot X$ を求めて1桁シフト・アップした部分剰余である．漸化式(4.4)における商数字 q_i の選択手順は，一意ではなく，このときの選択規則によって，様々なアルゴリズムが考えられるが，ここでは，$q_i\in\{\bar{1},0,1\}$ の場合を考える．そして，漸化式(4.4)を冗長2進加算で求めることにする．ただし，加数 $w=[w_0.w_1\cdots w_{n-1}]_2=-q_i\cdot X$ に関しては $\{0,1\}$ の表現に制限し，負の数は2の補数表現で扱うことにする．すなわち，

$$w=-w_0\cdot 2^0+\sum_{i=1}^{n-1}w_i\cdot 2^{-i} \tag{4.5}$$

である．そして，符号の変換は，

$$-w=-(1-w_0)\cdot 2^0+\sum_{i=1}^{n-1}(1-w_i)\cdot 2^{-i}+2^{-n+1} \tag{4.6}$$

によって行われる．被加数（部分剰余）は $r=[r_0.r_1\cdots r_{n-1}]_{RB2}$ のように冗長2進数表現で扱う．このように加数は $\{0,1\}$ で，被加数は $\{\bar{1},0,1\}$ で表現し，冗長2進加算することによって演算規則を簡単化でき

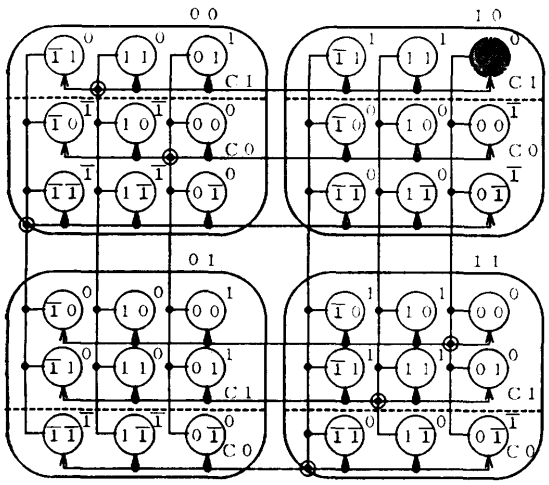


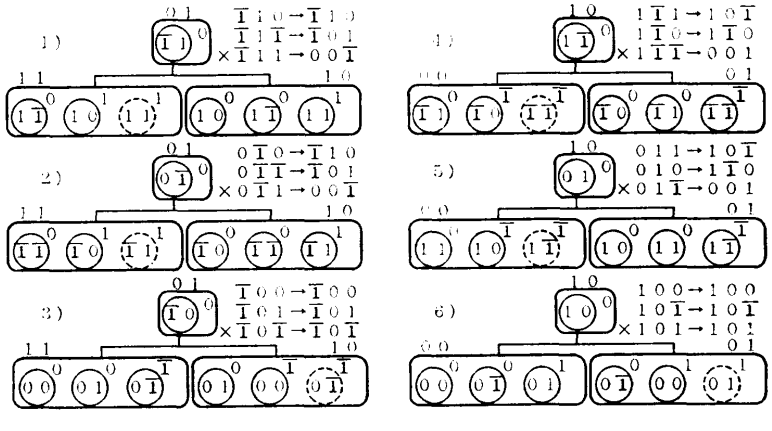
図 4 除算，開平アルゴリズムで採用する演算規則のストリング・グラフ表現
Fig. 4 String graph representation for computation rule introducing divide and square algorithms.

る．その演算規則をストリング・グラフ表現で示すと**図 4** のようになる．図 4 では，図を簡略化する工夫として図 3 とは逆に内側のノードに被加数を，外側の高階ノードに加数を割り当てている．また，被加数ノードには，後の解析に便利なように演算結果をラベル付けしている．図 4 の表現は，加数を引くために符号変換をした後の扱いを表現するにも極めて都合がよい．なぜならば，符号変換は，式(4.6)から，各桁の値を反転して，最下位桁を＋1するのであるが，これは最下位桁への桁上げ入力として扱えるからである．また，加数の最上位第0桁は，式(4.5)から $-w_0$ の意味であるが，第0桁の冗長2進加算は，桁上げ状態を無視できれば w_0 としても結果は同じなので，すなわち，桁上げがないことを後に保証するので，w_0 として扱うことができる．

4.2 ストリング・グラフ表現による商数字選択規則の導出と証明

図 4 の冗長2進加算規則のストリング・グラフ表現を利用して，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．まず，$q_0=1$ から出発し，漸化式(4.4)に従って冗長2進加算し，部分剰余 R_{i+1} を求める．ただし，各演算ステップごとに1桁シフト・アップするので，漸化式(4.4)を冗長2進加算で求めるためには，シフト・アップ前は常に最上位桁 r_0 がゼロになることを保証しなければならない．そのためには，まず，$1>Y=[0.1\ y_2\cdots]_2\geqq 2^{-1}$，$1>X=[0.1\ x_2\cdots]_2\geqq 2^{-1}$ を仮定する．これは正規化された2進浮動小数点数の場合には保証されている．

図 4 のストリング・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，有向辺を逆に見てたどる．そして，$q_i=1$ を選択するときは，加数は $[1.0\cdots]_2(=-X)$，$q_i=\bar{1}$ を選択するときは，加数は $[0.1\cdots]_2(=X)$ のかたちに表現されるので，**図 5** に示すように，それぞれ(b)加数ノード 10 と(a)加数ノード 01 から出発して上位3桁分をストリング・グラフ展開してみる．部分剰余のシフト・アップ前の結果が常に0…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．すなわち，ラベル値が常に0でない被加数ノード 11，$\bar{1}\bar{1}$ と 00 を除外する．これは，$1>Y=[0.1\ y_2\cdots]_2\geqq 2^{-1}$ という仮定から許される．そして，次の演算ステップ以降では，まず，被加数ノード 00 が現れた場合については，$q_i=0$ を選択し，そのまま1桁分シフト・アップすればよい．そして，残る被加数ノード 11 と $\bar{1}\bar{1}$ については，次の演算ステップでも現れるのを避けるため



(a) 加数ノード 01　　(b) 加数ノード 10

結果が 11 または $\bar{1}\bar{1}$ となるため，禁止される（×）被加数ノード

図 5 q_i の選択規則を求めるために，図4のストリング・グラフを上位3桁分展開したもの
Fig. 5 Development of string graph in Fig. 4 which is upper part of three digits in order to obtain selection rule of q_i.

表 1 剰余 r の上位3桁 $r_0r_1r_2$ の書換え変換 (d: don't care)
Table 1 Rewriting of upper part of three digits $r_0r_1r_2$ in remainder r.

入力	出力	入力	出力	入力	出力
$r_0\ r_1\ r_2$	$r_0'\ r_1'\ r_2'$	$r_0\ r_1\ r_2$	$r_0'\ r_1'\ r_2'$	$r_0\ r_1\ r_2$	$r_0'\ r_1'\ r_2'$
$\bar{1}\ \bar{1}\ \bar{1}$	d	$0\ \bar{1}\ \bar{1}$	$\bar{1}\ 0\ 1$	$1\ \bar{1}\ \bar{1}$	$0\ 0\ 1$
$\bar{1}\ \bar{1}\ 0$	d	$0\ \bar{1}\ 0$	$\bar{1}\ 1\ 0$	$1\ \bar{1}\ 0$	$1\ \bar{1}\ 0$
$\bar{1}\ \bar{1}\ 1$	d	$0\ \bar{1}\ 1$	$0\ 0\ \bar{1}$	$1\ \bar{1}\ 1$	$1\ 0\ \bar{1}$
$\bar{1}\ 0\ \bar{1}$	$\bar{1}\ 0\ \bar{1}$	$0\ 0\ \bar{1}$	$0\ 0\ \bar{1}$	$1\ 0\ \bar{1}$	$1\ 0\ \bar{1}$
$\bar{1}\ 0\ 0$	$\bar{1}\ 0\ 0$	$0\ 0\ 0$	$0\ 0\ 0$	$1\ 0\ 0$	$1\ 0\ 0$
$\bar{1}\ 0\ 1$	$\bar{1}\ 0\ 1$	$0\ 0\ 1$	$0\ 0\ 1$	$1\ 0\ 1$	$1\ 0\ 1$
$\bar{1}\ 1\ \bar{1}$	$\bar{1}\ 0\ 1$	$0\ 1\ \bar{1}$	$0\ 0\ 1$	$1\ 1\ \bar{1}$	d
$\bar{1}\ 1\ 0$	$\bar{1}\ 1\ 0$	$0\ 1\ 0$	$1\ \bar{1}\ 0$	$1\ 1\ 0$	d
$\bar{1}\ 1\ 1$	$0\ 0\ \bar{1}$	$0\ 1\ 1$	$1\ 0\ \bar{1}$	$1\ 1\ 1$	d

に，図5に示すように，現在の被加数の上位3桁から $1\bar{1}\bar{1}$, $01\bar{1}$, 101, $\bar{1}11$, $0\bar{1}1$, $\bar{1}0\bar{1}$ が現れないようにしなければならない．そこで，部分剰余を求めてシフト・アップ後，被加数の上位3桁に関して，図5または**表1**に示すような書換え変換を行う．この書換え変換は上記パターンが現れないように冗長2進数がもつ意味範囲内で行っているが，さらに上位1桁で商数字を決定することも考慮している．その結果，まず，被加数 101 と $\bar{1}0\bar{1}$ 以外が消去される．それで，以下では，被加数 101 と $\bar{1}0\bar{1}$ が現れても問題がないことを示す．まず，図5の3) と6) から，加数が 100 または 011 が選ばれる場合（図の左側）は問題ないので，加数が 101 または 010 が選ばれる場合（図の右側）のみについて考える．0ステップ目では，仮定から被加数は 011 または 010 なので，シフト・アップ後の結果は必ず0…になる．そこで，書換え変換しても，$\bar{1}0\bar{1}$, $\bar{1}00$, 101, 100 は現れない．そして，任意の i ステップ目でも，シフト・アップ後の結果が $\bar{1}0\bar{1}$ または $\bar{1}00$ となる場合は存在しない．なぜならば，図の4) または5) の場合，それぞれ対応する被加数 $1\bar{1}\bar{1}$, $01\bar{1}$ は禁止されていて，既に書換え変換で消去されているからである．そのまま1桁分シフト・アップする場合（図の1と2）でも，シフト・アップ後は $0\bar{1}\cdot$ であるから書換え変換によって $\bar{1}0\bar{1}$ または $\bar{1}00$ となる場合は存在しない．同様の理由によって，シフト・アップ後の結果が 101 または 100 となる場合（図の1，2，4，5）も存在しない．以上の考察により，問題がないことがわかったので，結果をまとめると，

[除算アルゴリズム]

部分剰余の上位3桁 $r_0r_1r_2$ を表1に従って書換え変換し，その結果の上位1桁 r_0' によって，q_i の選択を以下のように決定する．

（1） 上位1桁 r_0' が1のとき，$q_i=1$

（2） 上位1桁 r_0' が $\bar{1}$ のとき，$q_i=\bar{1}$

（3） 上位1桁 r_0' が0のとき，$q_i=0$

を選択する．すなわち，書換え変換結果の上位1桁 r_0' の値をそのまま q_i とする．

このように各部分商を求める段階において，表1に示す書換え変換に従って上位1桁の値をそのままその桁の商数字とすることができる．表1に示された書換え変換を2値論理回路で実現するために，まず，3値論理で扱うことによって論理式の簡約化を考える．そこで，2値論理の簡約化のためによく使われるカルノー図を**図6**に示すように3値論理に拡張する．非ゼロの出力値に注目する．すなわち，出力値 $\bar{1}$ と 1 に関して，それぞれ ▨ と ▩ のように分類する．ここで d (don't care) を都合のよい方の値に解釈して分類に含める．そして，$x_i\in\{\bar{1},0,1\}$ $(i=0,1,2)$ によって x_i が真であることを表すようにすれば，以下の簡約された論理式が得られる．ここで，¬は論理否定記号である．

$$r_0'\begin{cases}\bar{1}_0'=\bar{1}_0\cdot\neg(1_1\cdot 1_2)+0_0\cdot\bar{1}_1\cdot\neg 1_2\\ 1_0'=1_0\cdot\neg(\bar{1}_1\cdot\bar{1}_2)+0_0\cdot 1_1\cdot\neg\bar{1}_2\end{cases}$$

$$r_1'\begin{cases}\bar{1}_1'=(1_0\cdot\bar{1}_1+0_0\cdot 1_1)\cdot 0_2\\ 1_1'=(\bar{1}_0\cdot 1_1+0_0\cdot\bar{1}_1)\cdot 0_2\end{cases}$$

$$r_2'\begin{cases}\bar{1}_2'=\neg 0_1\cdot 1_2+0_1\cdot\bar{1}_2\\ 1_2'=\neg 0_1\cdot\bar{1}_2+0_1\cdot 1_2\end{cases}\qquad(4.7)$$

式(4.7)を2値論理に直して論理回路を実現すると，

r_0 \ r_1r_2	$\bar{1}\bar{1}$	$\bar{1}0$	$\bar{1}1$	01	00	$0\bar{1}$	$1\bar{1}$	10	11	
$\bar{1}$	d	d	d	$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$	$\bar{1}$	0	
0	$\bar{1}$	$\bar{1}$	0	0	0	0	0	1	1	r_0'
1	0	1	1	1	1	1	d	d	d	
$\bar{1}$	d	d	d	0	0	0	0	1	0	
0	0	1	0	0	0	0	0	$\bar{1}$	0	r_1'
1	0	$\bar{1}$	0	0	0	0	d	d	d	
$\bar{1}$	d	d	d	1	0	$\bar{1}$	1	0	$\bar{1}$	
0	1	0	$\bar{1}$	1	0	$\bar{1}$	1	0	$\bar{1}$	r_2'
1	1	0	$\bar{1}$	1	0	$\bar{1}$	d	d	d	

図 6 3値論理の拡張カルノー図
Fig. 6 Modified Karnaugh map for three-valued logic.

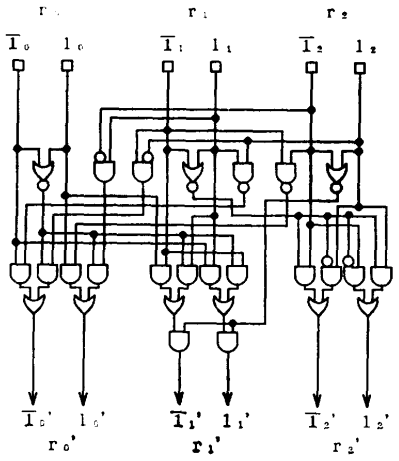


図 7 剰余 r の上位3桁 $r_0r_1r_2$ の書換え変換回路：出力 $r_0'=(\bar{1}_0', 1_0')$ が商数字を決定する.
Fig. 7 Rewriting circuit of upper part of three digits $r_0r_1r_2$: output $r_0'=(\bar{1}_0', 1_0')$ determines the quotient digit.

図7に示すように，剰余 r の上位3桁 $r_0r_1r_2$ の書換え変換回路を得ることができる．特に，出力 $r_0'=(\bar{1}_0', 1_0')$ が商数字を決定する.

5. おわりに

加減乗除などの種々の算術演算アルゴリズムに関して，それらのストリング・グラフ表現法を提案した．特に，除算アルゴリズムに関して，その導出と証明に応用した．今後は，基数4以上の除算アルゴリズムを具体的に構成するために応用することを考えている．冗長2進加算アルゴリズムそれ自身に関しては，そのストリング・グラフ表現を使ってさらに詳細に解析しているので，別途報告する.

謝辞 日頃の研究にご理解をいただいている当社中央研究所西向井忠彦主任研究員に感謝します.

参考文献

1) Hwang, K. (堀越 彌監訳)：コンピュータの高速演算方式，近代科学社，pp. 129-162 (1980).
2) Avizienis, A.: Signed Digit Number Representations for Fast Parallel Arithmetic, *IRE Trans. Electronic Comput.*, Vol. EC-10, No. 9, pp. 389-400 (1961).
3) 高木直史，安浦寛人，矢島脩三：冗長2進加算木を用いた VLSI 向き高速乗算器，信学論 (D), Vol. J 66-D, No. 6, pp. 683-690 (1983).
4) Takagi, N., Yasuhara, H. and Yajima, S.: High-Speed VLSI Multiplication Algorithm with a Redundant Binary Addition Tree, *IEEE Trans. Comput.*, Vol. C-34, No. 9, pp. 789-796 (1985).
5) 高木直史，安浦寛人，矢島脩三：冗長2進表現を利用した VLSI 向き高速除算器，信学論 (D), Vol. J 67-D, No. 4, pp. 450-457 (1984).
6) Harata, Y., Nakamura, Y., Nagase, H., Takigawa, M. and Takagi, N.: High Speed Multiplier Using a Redundant Binary Adder Tree, *Proc. of ICCD '84*, pp. 165-170 (1984).
7) 原田義久，服部佳晋，長瀬 宏，瀧川光治：冗長2進加算セルの回路構成の検討，電子情報通信学会全国大会予稿集，p. 2-82 (1987).
8) Rubinfield, L. P.: A Proof of the Modified Booth's Algorithm for Multiplication, *IEEE Trans. Comput.*, Vol. C-24, No. 10, pp. 1014-1015 (1975).
9) Craver, J. E. and Watkins, M. E.: *Combinatorics with Emphasis on the Theory of Graphs*, Springer-Verlag (1977).
10) Berge, C. (伊理正夫ほか訳)：グラフの理論Ⅲ，サイエンス社 (1976).
11) Nasu, M.: Local Maps Inducing Surjective Global Maps of One-Dimensional Tessellation Automata, *Mathematical System Theory*, Vol. 11, pp. 327-351 (1978).
12) Nakanishi, T., Yamauchi, H. and Yoshimura, H.: CMOS Radix-2 Signed-Digit Adder by Binary Code Representation, *Trans. IECE Japan*, Vol. E69, No. 4, pp. 261-263 (1986).
13) 仲西 正：2進冗長 SD コードの2値符号化方式，公開特許公報 (A), 昭 62-204332 (1987).
14) Kuninobu, S., Nishiyama, T., Edamatsu, H., Taniguchi, T. and Takagi, N.: Design of High Speed MOS Multiplier and Divider Using Redundant Binary Representation, *IEEE International Symposium on Computer Arithmetic*, pp. 80-86 (1987).
15) 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，ICD 87-100, pp. 43-48 (1988).
16) 高木直史，西山 保，國信茂郎：演算処理装置，公開特許公報 (A), 昭 63-25728-25729, 昭 63-49835-49836 (1988).
17) Edamatsu, H., Taniguchi, T., Nishiyama, T. and Kuninobu, S.: A 33 MFlops Floating Point Processor Using Redundant Binary Representation, *Proc. of ISSCC 88*, pp. 152-153 (1988).
18) 高木直史，矢島脩三：冗長2進表現を利用した開平用ハードウェアアルゴリズム，信学論 (D), Vol. J 69-D, No. 1, pp. 1-10 (1986).
19) De Bruijn, N. G.: A Combinatorial Problem, *Proc. Kon. Ned. Akad. Wetensch*, Vol. 49, pp. 758-764 (1946).
20) Lempel, A.: On a Homomorphism of the de Bruijn Graph and Its Applications to the

Design of Feedback Shift Registers, *IEEE Trans. Comput.*, Vol. C-19, No. 12, pp. 1204-1209 (1970).
21) Esfahanian, A. H. and Hakimi, S. L.: Fault-Tolerant Routing in De Bruijn Communication Networks, *IEEE Trans. Comput.*, Vol. C-34, No. 9, pp. 777-788 (1985).
22) Sridhar, M. A.: On the Connectivity of the De Bruijn Graph, *Inf. Process. Lett.*, Vol. 27, pp. 315-318 (1988).
23) 外村元伸：演算アルゴリズムのストリング・グラフ表現，第 37 回情報処理学会全国大会論文集，(I) 2D-4, pp. 7-8 (1988).
24) Zurawski, J. H. P. and Gosling, J. B.: Design of High-Speed Square Root Multiply and Divide Unit, *IEEE Trans. Comput.*, Vol. C-36, No. 1, pp. 13-23 (1987).

(平成元年 11 月 8 日受付)
(平成 2 年 4 月 17 日採録)

外村 元伸 (正会員)

1952 年生. 1979 年名古屋大学大学院工学研究科情報工学専攻博士課程前期課程修了. 同年，(株)日立製作所中央研究所入社. 制御用ミニコンピュータ，3 次元グラフィック・プロセッサ，GMICRO/200 の開発に従事. 現在，ULSI プロセッサ，特に演算・制御の論理・回路方式などの研究に従事. IEEE，日本応用数理学会各会員.