Vol. 25 No. 6 情報処理学会論文誌 Nov. 1984

キャリーセーブ方式による2の補数表示除算配列とその評価†

久津輪 敏郎†† 文 民 浩††

現在の演算方式や各種演算器をみると，加減算は2の補数表示であるが，乗除算とりわけ除算は符号と絶対値表示である．このような四則演算での数の表現の不統一は，演算の都度，表現を変換する必要があり，計算機の構成上不都合である．四則演算のすべてをどちらかの表現に統一するほうがよいと考えられる．この観点から本論文では，伝搬遅延時間を減少させる効果のあるキャリーセーブ方式に基づいて，引放し法および引戻し法による2の補数表示除算の原理を示し，商および剰余の補正を理論的に導く．そしてこれに基づく除算配列の構成を示し，両配列の遅延時間，ゲート数の比較を行い，引放し法の配列が引戻し法のものより優れていることを明らかにする．

1. まえがき

最近の集積回路技術の進歩に伴って，各種の並列型の演算器が発表されつつある．このうち加減算は2の補数表示を扱うが，乗除算とりわけ除算は絶対値表示の数（正の数）のみを扱うのがほとんどである．

このことは現在主流をなす計算機が，加減算は2の補数表示で演算するのに対し，乗除算は符号・絶対値表示で行っていることによると考えられる．しかしこのような四則演算での数の表現の不統一は計算機の構成上不都合である．すなわち，多項式や有理関数のように加減乗除が組み合わさった計算において，その都度符号・絶対値表示と2の補数表示の間で数の変換をしなければならず不便である．むしろ，四則演算のすべてをどちらかの表示に統一して行うほうがよいと考える．

この観点から本論文では，2の補数表示の数を扱う除算配列の構成について検討する．なお高速化をはかるためにキャリーセーブ方式を採用した引放し法，引戻し法による配列の構成を示し，遅延時間，ゲート数およびその積について評価を行う．

ゲート数を評価の対象とすることには異論があるかもしれない．VLSIでの回路の規模は素子（ゲート）数だけでなく配線や外部端子数も考慮に入れるべきだとして，その評価にも種々の提案がある[1]．しかし配線の面積への影響は素子の配置や線の引き方により大幅に異なるので，その算定基準はまだ明らかにされていない．現在の評価の仕方は，たとえば一つの配列は単位回路が1行につき n 個，それが $O(\log n)$ 行であるとき，面積は $O(n \log n)$ とみる☆．もし他の配列が $O(n^2)$ であれば前者が面積は小さいと判断する．この評価は単位回路自体が同種の素子でかつ同一の規模ならばよいが，そうでなければ意味をなさない．これらのことから，現時点ではまだ配線のような不確定要素を入れて評価をあいまいにするよりもゲート数で比較するほうがよいと考える．

以下，2章で引放し法の除算の原理，具体的演算の方法を示し，商および剰余の補正を理論的に導く．3章ではそれに基づく除算配列の構成を示す．4章では引戻し法の除算の原理および商・剰余の補正，5章ではそれによる除算配列を示し，6章では両配列および絶対値表示除算配列についてゲート数，遅延時間の比較を行う．

2. 引放し法による除算

2.1 除算の原理

引放し法は一般に2の補数表示の数の除算を行えるといわれるが[2]，現在までの除算配列では（正）÷（正）以外は正しい解が得られない[3]．

2の補数表示の被除数 D，除数 V を次のように表す．

$$D=(d_0\ d_1\ \cdots\ d_{n-1}\ \cdots\ d_{2n-1})$$
$$V=(v_0\ v_1\ \cdots\ v_{n-1})$$

ここで d_0，v_0 は符号ビットで0ならば正，1ならば負を表す．n はビット数を示す．

$d_0=v_0$ ならば $q_0'=1$，$d_0 \neq v_0$ ならば $q_0'=0$ とおき

† Configuration and Evaluation of 2's Complement Division Arrays Based on the Carry-Save Method by TOSHIRO KUTSUWA and MINHO MUN (Dept. of Electronics, Faculty of Engineering, Osaka Institute of Technology).

†† 大阪工業大学工学部電子工学科

☆ $O(n)$ は n のオーダーを示す．

表1 商の決定
Table 1 Determination of quotient bit.

区分	被除数 D	除数 V	桁上 c_{ai}	剰余 R_i	R_i と V の符号	商 q_i
(i)	正	正	0	負	異	0
			1	正	同	1
(ii)	正	負	0	負	同	1
			1	正	異	0
(iii)	負	正	0	負	異	0
			1	正	同	1
(iv)	負	負	0	負	同	1
			1	正	異	0

$$R_0=D+(1-2q_0')V \tag{1}$$

すなわち

$$R_0=\begin{cases}D-V & (q_0'=1)\\ D+V & (q_0'=0)\end{cases} \tag{1}'$$

を行う．そして R_0 と V の符号が等しければ商のビットを $q_0=1$，異なれば $q_0=0$ とする．

同様に $i=1,2,\cdots,n-1$ について，前の商ビット q_{i-1} の値に従って，

$$R_i=2R_{i-1}+(1-2q_{i-1})V \tag{2}$$

すなわち

$$R_i=\begin{cases}2R_{i-1}-V & (q_{i-1}=1)\\ 2R_{i-1}+V & (q_{i-1}=0)\end{cases} \tag{2}'$$

を行い，R_i と V が同符号なら $q_i=1$，異符号ならば $q_i=0$ とする．なお最終行の剰余は $|R_{n-1}|\leq|D|$ であるが，符号は簡単な規則で決まらない[2]．このため商の最下位ビットおよび剰余に対し補正が必要である．補正の方法は2.3節で述べる．

2.2 具体的演算

式(1)，(2)の計算における R_i，V の符号の判定および商の決定は，最上位ビットからの桁上 c_{ai} により，表1のように行う．表1の第1行は (正)÷(正) において桁上 c_{ai} が0であれば，剰余 R_i は負で剰余と除数は異符号となるから，商ビット q_i は0となることを示す．この表から

$$q_i=v_0\oplus c_{ai} \tag{3}$$

が得られる．

2.3 商および剰余の補正

本節では，これらの補正の方法を理論的に導く．なお補正の仕方には，剰余の符号を，(i)被除数の符号に一致させる，(ii)除数の符号に一致させる，(iii)つねに正にする，などが考えられるが，ここでは(i)を採用している．

いま第 $n-1$ 行まで割り切れないとするとき，第 n 行の剰余 R_n は

$$R_n=2R_{n-1}+(1-2q_{n-1})V \tag{4}$$

である．この式に式(2)を代入すると

$$\begin{aligned}R_n&=2(2R_{n-2}+(1-2q_{n-2})V)+(1-2q_{n-1})V\\&=2^nR_0+\sum_{i=1}^{n}2^{n-i}(1-2q_{i-1})V\\&=2^n\Big[R_0+\Big(\sum_{i=1}^{n}2^{-i}-\sum_{i=1}^{n}2^{-i+1}q_{i-1}\Big)V\Big]\\&=2^n\Big[D+\Big\{(1-2q_0')+\sum_{i=1}^{n}2^{-i}\\&\quad-\sum_{i=1}^{n}2^{-i+1}q_{i-1}\Big\}V\Big]\end{aligned} \tag{5}$$

したがって，式(4)より第 $n-1$ 行の剰余は

$$\begin{aligned}R_{n-1}&=2^{n-1}\Big\{D+(1-2q_0')V+\Big(\sum_{i=1}^{n}2^{-i}\\&\quad-\sum_{i=1}^{n}2^{-i+1}q_{i-1}\Big)V\Big\}-2^{-1}(1-2q_{n-1})V\\&=2^{n-1}\Big[D+\Big\{2(1-q_0')-\sum_{i=1}^{n}2^{-i+1}q_{i-1}\\&\quad-2^{-n+1}+2^{-n+1}q_{n-1}\Big\}V\Big]\end{aligned} \tag{6}$$

となる．ゆえに D は

$$\begin{aligned}D=\Big[&2(q_0'-1)+\sum_{i=1}^{n}2^{-i+1}q_{i-1}+2^{-n+1}\\&-2^{-n+1}q_{n-1}\Big]V+2^{-n+1}R_{n-1}\end{aligned} \tag{7}$$

となる．ここで $2(q_0'-1)$ は符号ビットより上の桁になるので無視すると D は次のようになる．

$$\begin{aligned}D=&\Big[\sum_{i=1}^{n}2^{-i+1}q_{i-1}\Big]V\\&+[2^{-n+1}\{R_{n-1}+(1-q_{n-1})V\}]\end{aligned} \tag{8}$$

または

$$\begin{aligned}D=&\Big[\sum_{i=1}^{n}2^{-i+1}q_{i-1}+2^{-n+1}\Big]V\\&+[2^{-n+1}(R_{n-1}-q_{n-1}V)]\end{aligned} \tag{9}$$

これらの式は $D=QV+R$ であり，右辺第1項の [ ] が商 Q，第2項が剰余 R を示す．

これらの式によって商および剰余の補正を決定することができ，それを表2に示す．

まず最終の $n-1$ 行まで割り切れない（後述の割切判定出力 h が0である）場合を考える．

たとえば (正)÷(正) で第 $n-1$ 行の商ビット q_{n-1} が0ならば剰余 R_{n-1} は負であり，D と R_{n-1} は異符号である．剰余 R を正にして D の符号と合わせるに

表 2 剰余 R および商 Q の補正
Table 2 Corrections of remainder and quotient.

行番号	被除数 D	除数 V	商（最下位）q_{n-1}	割切判定 h	Rの補正 $\times 2^{-n+1}$	Qの補正 $\times 2^{-n+1}$
1	正	正	0	0	$+V$	
2				1	$+V$	
3			1	0		
4				1		
5	正	負	0	0		$+1$
6				1		$+1$
7			1	0	$-V$	$+1$
8				1	$-V$	$+1$
9	負	正	0	0		$+1$
10				1	$+V$	
11			1	0	$-V$	$+1$
12				1		
13	負	負	0	0	$+V$	
14				1		$+1$
15			1	0		
16				1	$-V$	$+1$

は，式(8)から

$$R=2^{-n+1}\{R_{n-1}+(1-q_{n-1})V\}$$
$$=2^{-n+1}\{R_{n-1}+V\}$$

とする．すなわち R_{n-1} に $+V$ を加えて補正しなければならない．これを表2の第1行に示す．$h=0$ の他の組合せについても同様に導ける．これらを表2の奇数 (3, 5, …, 15) 行に示す．

つぎに，途中の行 $l(0\leq l<n-1)$ で割り切れる ($h=1$ である) 場合を考える．行 l で割り切れるとき $c_{al}=1$, $R_l=0$ となるが，除算配列では最終の $n-1$ 行まで演算を続行し誤差を生ずるので補正が必要である．

(i) (正)÷(正) では $c_{al}=1$ により表1から $q_i=1$ であり，式(2)によって $i(l<i\leq n-1)$ に対して $R_i=-V<0$, $q_i=0$ となる．ゆえに式(8)から剰余は，R_{n-1} に $+V$ の補正が必要である．これを表2の第2行に示す．

(ii) (正)÷(負) では $c_{al}=1$ により $q_l=0$ であり，$i(l<i\leq n-1)$ に対して $R_i=+V<0$, $q_i=1$ となるから式(9)によって剰余は R_{n-1} に $-V$, 商は最下位ビット q_{n-1} に $+1$ の補正が必要である．これを表2の第8行に示す．(iii)の(負)÷(正)，(iv)の(負)÷(負) の場合も同様に導ける．これを表2の第 10, 16 行に示す．

(iii) (−)/(+)の場合

$D=1.111(=-0.001)$
$V=0.010$

	正答	結果
$Q=$	1.100	1.100
$R=$	0.000×2^{-3}	1.110×2^{-3}

Rに$+0.010(=V)\times 2^{-3}$の補正

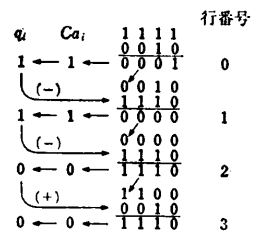

図 1 引放し法による2の補数表示除算の例
Fig. 1 2's complement division based on the non-restoring procedure.

2.4 計 算 例

具体的計算例を図1に示す．これは(iii)の(負)÷(正)でかつ途中（第1行）で割り切れる例である．

$d_0 \neq v_0$ であるから $q_0'=1$ とし加算を行う．桁上 $c_{a1}=1$ であるから $q_1=1$ とし，次の行は減算となる．なお，もし $c_{a1}=0$ なら $q_1=0$ とし次の行は加算となる．以下同様に行うと正答 $Q=1.100$, $R=0.000\times 2^{-3}$ に対して，結果は $Q=1.100$, $R=1.110\times 2^{-3}$ となる．この例は第1行で割り切れ $h=1$ であり，$q_{n-1}=0$ であるから，R_{n-1} に $+V$ の補正を行うとよいことがわかる．

3. 引放し法による除算配列

図2は，引放し法によって2の補数表示の除算を行う配列を示す．演算部の各セルを図2(b)に示す[4]．第0行は $d_0=v_0$ ならば減算，$d_0 \neq v_0$ ならば加算を行う．式(3)から $c_{ai}\oplus v_0$ によって商ビット q_i が決定する．$q_i=1$ ならば次の行では減算，$q_i=0$ ならば加算となる．

この配列はキャリーセーブ方式を採用しているので第 $n-1$ 行の剰余 R_{n-1} を求めるには，その行の各和出力 s_j^{n-1} とキャリー出力 c_j^{n-1} の加算を行わなければならない．これを実行するのが剰余加算部であり，n ビットのリップル型の加算器などで行う．

割切判定部は，2.3 節で述べたように最終の $n-1$ 行またはそれ以前に割り切れたか否かを判定する回路である．最終行で割り切れたときは剰余 R_{n-1} が0である．途中の行 $l(0\leq l<n-1)$ で割り切れた場合は前述のように R_{n-1} は $\pm V$ となっているので，$\mp V$ を

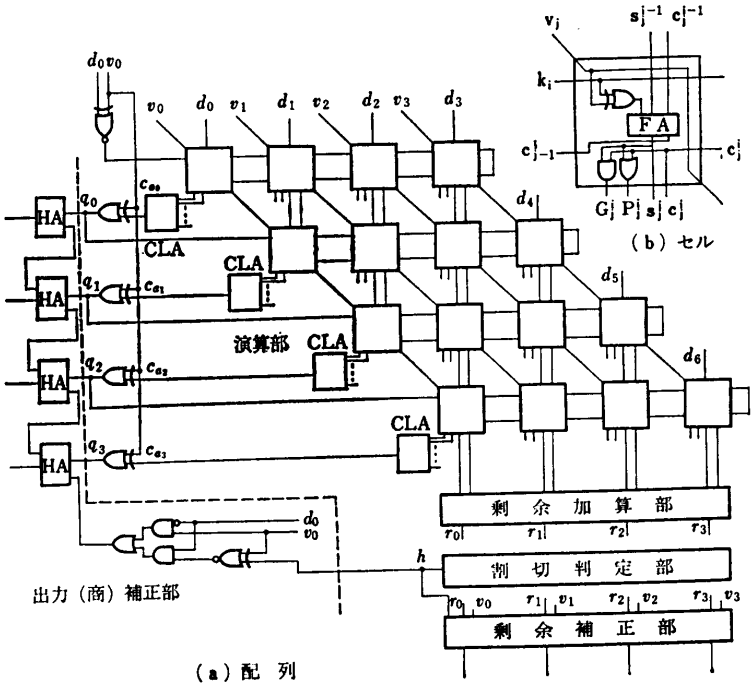

図 2　引放し法による2の補数表示除算配列 ($n=4$)
Fig. 2　2's complement division array based on the non-restoring procedure.

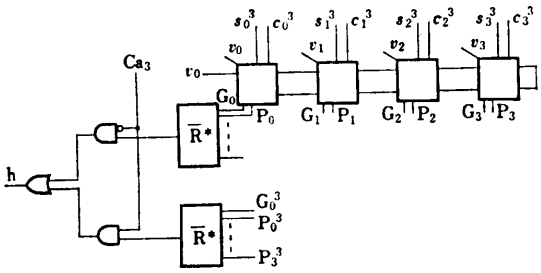

図 3　図2の割切判定部
Fig. 3　Divisible judging part of Fig. 2.

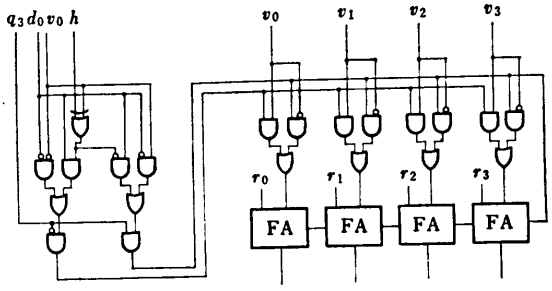

図 4　剰余補正部 ($n=4$)
Fig. 4　Remainder correcting part.

加えて和が0となるかを調べるかまたはこれと等価な働きをする**図3**の回路を用いればよい．図3の $\bar{R}^*$ は一般的につぎのように表される．

$$\bar{R}^*=\overline{G_0(P_1+P_2+\cdots+P_{n-1})+G_1(P_2+P_3+\cdots P_{n-1})+\cdots+G_{n-2}P_{n-1}} \tag{10}$$

図3の上位の $\bar{R}^*$ は途中の行で割り切れるか否かの判定である．下位の $\bar{R}^*$ は最終行の割切判定であり，この場合には式(10)の $G_j, P_j (j=1,\cdots,n-1)$ のかわりに，$n-1(=3)$ 行のセルの G_j^3, P_j^3 を入力する．

途中または最終行で割り切れたとき，割切判定出力 h は1となる．

商および剰余の補正回路は表2の真理値表によって構成すればよい．**図4**に剰余補正の回路を示し，商の補正は図2の最左列における n 個の半加算器 HA などによって行う．

4. 引戻し法による除算

4.1 除算の原理

(i)　(正)÷(正)

被除数 X_0 ($=D>0$) および2の補数化した除数 B ($=\tilde{V}<0$)* を演算部へ入力し，$i=0,1,\cdots,n-1$ について次の演算を行う

$$R_i=X_i+B \tag{11}$$

もし最上位からの桁上 c_{ai} が0で剰余が $R_i<0$ ならば，中間結果 $k_i=0$，商ビット $q_i=0$ とおき，$X_{i+1}=2X_i$ とする．$c_{ai}=1$ すなわち $R_i\geq 0$ ならば，$k_i=$

* V は V の2の補数を意味する．

1, $q_i=1$ とおき, $X_{i+1}=2R_i$ とする.

(ii) (正)÷(負)

被除数 $X_0(=D\geq 0)$, 除数 $B(=V<0)$ をそのまま演算部へ入力し, (i)と同様の演算を行う. ただし商 $Q=(q_0\ q_1\ \cdots\ q_{n-1})$ を得るには, 中間結果 $K=(k_0\ k_1\ \cdots\ k_{n-1})$ を2の補数化する必要がある. そこで各 k_i を $0\leftrightarrow 1$ に反転し, 最下位ビットに $+1$ を行う.

(iii) (負)÷(正)

被除数 $X_0(=D\leq 0)$, 除数 $B(=V>0)$ をそのまま演算部へ入力し, 式(11)の演算を行う. もし $c_{ai}=1$ すなわち $R_i\geq 0$ ならば, $k_i=0$, $q_i=1$ とおき. $X_{i+1}=2X_i$ とする. $c_{ai}=0$ すなわち $R_i<0$ ならば $k_i=1$, $q_i=0$ とおき $X_{i+1}=2R_i$ とする. そして, 最終の $n-1$ 行まで割り切れないならば商 Q の最下位に $+1$ して2の補数を得る.

(iv) (負)÷(負)

被除数 $X_0(=D\leq 0)$, 2の補数化した除数 $B(=\bar{V}\leq 0)$ を演算部へ入力する. $c_{ai}=1$ すなわち $R_i\geq 0$ ならば $k_i=0$, $q_i=0$ とおき $X_{i+1}=2X_i$ とする. $c_{ai}=0$ すなわち $R_i<0$ ならば, $k_i=1$, $q_i=1$ とおき $X_{i+1}=2R_i$ とする.

4.2 商および剰余の補正

(i), (ii)の場合は最終行までに割り切れても, $R_{n-1}=0$ となるので補正を必要としないが, (iii), (iv)では最終行までに割り切れるとき, 別の補正が必要である.

いま $l(0\leq l\leq n-1)$ 行で割り切れると仮定する. (iv)では, 割り切れて $R_l=0$ のときに $c_{al}=1$ で, $k_l=0$, $q_l=0$ となり, $X_{l+1}=2X_l=-2B=2V<0$ となる. このため $i>l$ に対し, $c_{ai}=0$, $k_i=1$, $q_i=1$, $R_i=-B=V<0$ となる. 実際に l 行で割り切れているのならば, $k_l=q_l=1$ とし, $i>l$ に対し $k_i=q_i=0$ に補正する必要があるので $q_{n-1}(=k_{n-1})$ に $+1$ を行って $q_l=1$ にし, さらに剰余 $R(R_{n-1})$ を0に設定する.

(iii)の(負)÷(正)は, (iv)の(負)÷(負)で得られる中間結果 $K=(k_0\ k_1\ \cdots\ k_{n-1})=(q_0\ q_1\ \cdots\ q_{n-1})$ を2の補数化したものが商 $Q=(q_0\ q_1\ \cdots\ q_{n-1})$ であるが, l 行で割り切れて $R_l=0$ のとき, $q_l=1$, $i>l$ に対し $q_i=0$ となり $R_i=-B=-V<0$ となっているから商には補正が必要でなく, 剰余 $R(R_{n-1})$ のみを0に設定すればよい.

割切れの判定は, 引放し法の場合と同様に $-V$ または $+V$ を加えて剰余が0となるかをみればよい.

(iii) (−)/(+)の場合

$D=1.111(=-0.001)$
$V=0.010$

正答	結果
$Q=1.100$	1.100
$R=0.000\times 2^{-3}$	1.110×2^{-3}

Rに$+0.010(=V)\times 2^{-3}$の補正

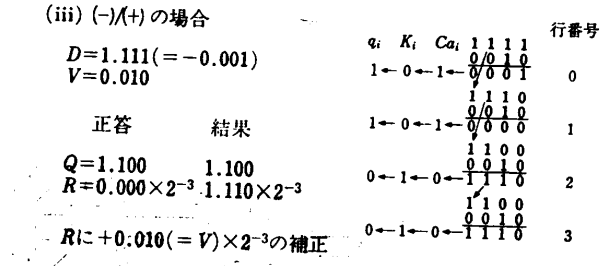

図5 引戻し法による2の補数表示除算の例
Fig. 5 2's complement division based on the restoring procedure.

4.3 計算例

2.4節と同じ例について, 引戻し法によって計算を行うと図5のようになる. すなわち $d_0\neq v_0$ であるからそのまま入力し加算を行う. $c_{ai}=1$ で $k_i=0$, $q_i=1$ とおき $X_{i+1}=2X_i$ すなわち X_i を左へシフトする. もし $c_{ai}=0$ ならば $k_i=1$, $q_i=0$ とおき $X_{i+1}=2R_i$ すなわち R_i を左へシフトする. 以下同様に行うと正答 $Q=1.100$, $R=0.000$ であるのに対し, 結果は $Q=1.100$, $R=1.110\times 2^{-3}$ となる. この例は第1行で割り切れる場合であるから, 4.2節で述べたように剰余に $+V$ を行って割り切れることを確かめ, 剰余を0に設定する.

5. 引戻し法による除算配列

図6は引戻し法による2の補数表示除算配列を示す. 入力補正部においては, $d_0=v_0$ のときに除数 V が2の補数に変換される. 同図(b)に示される演算部のセルは Agrawal[5] のものと同じであり, その論理動作は次式となる.

$$s_j^i=(s_j^{i-1}\oplus c_j^{i-1})\oplus(k_i b_j) \tag{12}$$

$$c_{j-1}^i=s_j^{i-1}c_j^{i-1}+(s_j^{i-1}+c_j^{i-1})k_i b_j \tag{13}$$

$$e_{j-1}^i=s_j^{i-1}c_j^{i-1}+(s_j^{i-1}+c_j^{i-1})b_j \tag{14}$$

$$G_j^i=(s_j^{i-1}\oplus b_j\oplus c_j^{i-1})e_j \tag{15}$$

$$P_j^i=s_j^{i-1}\oplus b_j\oplus c_j^{i-1}+e_j^i \tag{16}$$

s_j^{i-1} と c_j^{i-1} の和が被除数のビット x_j^{i-1} に等しい. s_j^i, c_{j-1}^i, e_{j-1}^i, G_j^i, p_j^{i-1} は, それぞれ和, 桁上, 予測桁上, CLA 用の桁上発生, 桁上伝搬の出力である. この配列(引戻し法のキャリーセーブ方式)の配列は最上位からの桁上 c_{ai} が明確に現れないので, 中間結果 k_i は図6の s, c_1, c_2, c_3 の論理値から次のように決定しなければならない[5].

$$k_i=\begin{cases}\bar{d}_0 c_{3i}+d_0(c_{1i}\oplus c_{3i}) & (i=0) \\ s_i\oplus c_{1i}\oplus c_{2i}\oplus c_{3i} & (i=1,2,\cdots,n-1)\end{cases} \tag{17}$$

(18)

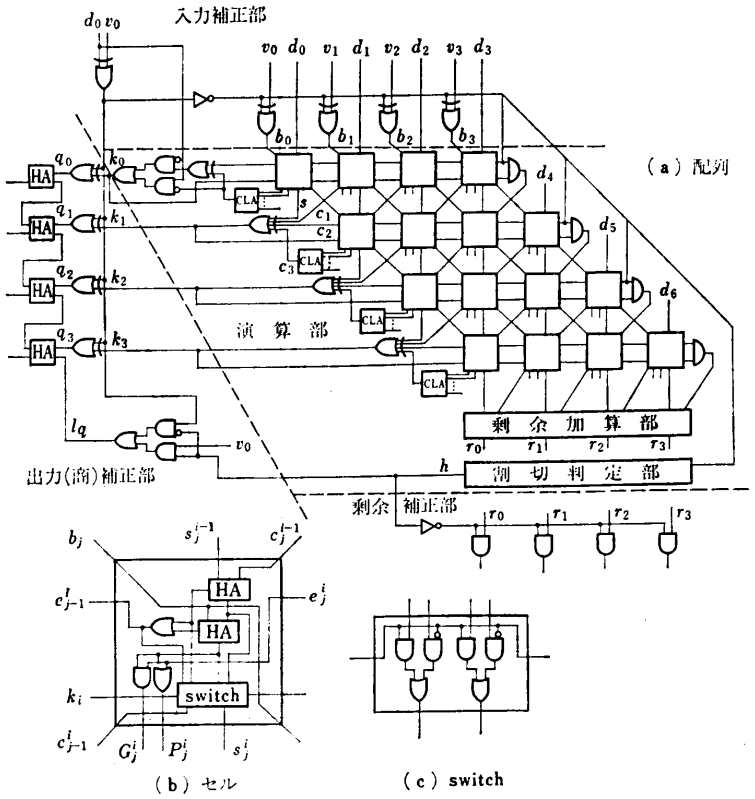

図 6　引戻し法による2の補数表示除算配列 ($n=4$)
Fig. 6　2's complement division array based on the restoring procedure.

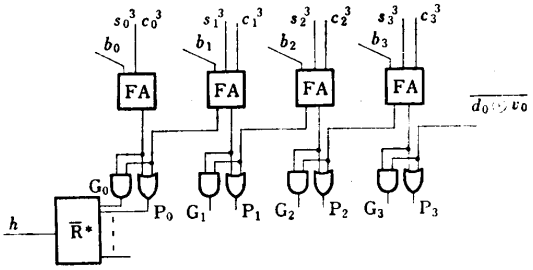

図 7　図6の割切判定部
Fig. 7　Divisible judging part of Fig. 6.

商のビット q_i，最下位ビットへの加算 $(+1)l_q$ は，

$$q_i=k_i\oplus(d_0\oplus v_0) \tag{19}$$

$$l_q=(d_0\oplus v_0)\bar{h}+v_0h \tag{20}$$

となる．

引戻し法の割切判定部は，**図7**のようになり，引放し法より回路は少し簡単になる．

また，剰余補正部は図6のように割切判定出力 h が1の時に剰余を0にセットする回路となる．

6. 両配列および絶対値表示除算配列の比較

引放し法，引戻し法による2の補数表示除算配列および絶対値表示除算配列についてゲート数および遅延時間を算出し，比較を行う．ここで絶対値表示の配列は，その入力および出力部に補数回路を付加して，2の補数表示に適用したものである．

6.1 ゲート数

いま，NOT を 1，2 入力 NAND, NOR を 2，2入力 AND, OR を 3，2 入力 XOR を8，半加算器 HA を 11，全加算器 FA を 28 とカウント* すれば，各配列のゲート数は**表3**のようになる．

表 3　配列に必要なゲート数
Table 3　Gates required to the arrays.

引放し配列	2の補数表示	$42n^2+124n+60+C_n^*\cdot n+2\cdot R_n^*$
	絶対値表示	$42n^2+156n-48+C_n^*\cdot n$
引戻し配列	2の補数表示	$45n^2+119n+9+C_n^*\cdot n+R_n^*$
	絶対値表示	$45n^2+147n-65+C_n^*\cdot n$

n: ビット数 (n は8の倍数)
C_n^*: n ビットの CLA 回路に必要なゲート数
R_n^*: n ビットの R^* 回路に必要なゲート数

* CMOS で NOT を基準にして換算している．

表 4 セルの遅延時間
Table 4 Propagation delay of a cell.
(単位：Δ 時間)

		入力						入力				
		v_j	s_j^{i-1}	c_j^{i-1}	k_i			b_j	s_j^{i-1}	c_j^{i-1}	k_i	e_j^i
出力	c_{j-1}^i	5	2	2	5	出力	e_{j-1}^i	4	4	4	—	—
	S_j^i	6	3	3	6		c_{j-1}^i	6	4	6	3	—
	G_j^i	8	5	5	8		s_j^i	5	8	8	3	—
	P_j^i	8	5	5	8		G_j^i	5	8	8	—	2
							P_j^i	5	8	8	—	2

表 5 配列の遅延時間
Table 5 Propagation delay of the arrays.
(単位：Δ 時間)

		$n \geq 24$	$n \leq 16$
引放し配列	2の補数表示	$19n+21$	$17n+19$
	絶対値表示	$24n+8$	$22n+8$
引戻し配列	2の補数表示	$26n+23$	$24n+19$
	絶対値表示	$30n+7$	$28n+7$

n：ビット数（nは8の倍数）

ゲート数に関しては，2の補数表示の配列が絶対値表示のそれよりもやや少ない．

6.2 遅延時間

NOT，NAND，NORが1個あたり1Δ，AND，ORは2Δの遅延時間をもつとする[6]．ただし2レベルAND-ORはNAND-NANDとみて2Δとする．Δはゲートの単位遅延時間を表す*．

演算部のセルの遅延は表4のようになる．したがって各配列の遅延は表5のようになる．各配列の最長伝搬経路は，2の補数表示の引放し配列が符号比較から剰余補正部へ，引戻し配列が入力補正部から出力(商)補正部へ，絶対値表示では入力の符号変換から剰余の符号変換へ至る経路である．

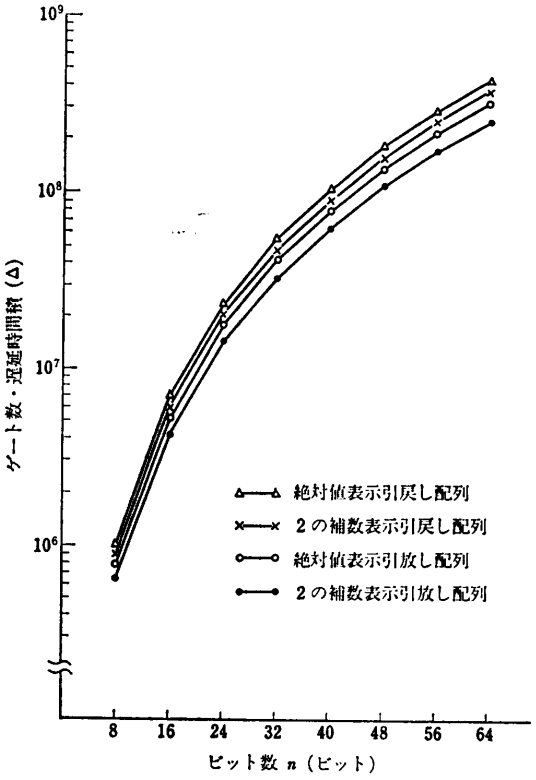

図 8 配列のゲート数・遅延時間積
Fig. 8 The product of gate by propagation delay of arrays.

6.3 評価

LSIの評価としては一般に面積と遅延時間の積とされているが，本文の各配列は同等の規則性をもっているのでここでは先に述べたようにゲート数と遅延時間の積をとる．これを表6に示す．そしてこれをグラフに表わせば図8が得られる．ゲート数，遅延時間とも小

表 6 配列のゲート数・遅延時間積
Table 6 The product of gate by propagation delay of the arrays.

ビット数	引放し配列		引戻し配列	
n	2の補数表示	絶対値表示	2の補数表示	絶対値表示
8	6.41×10^5	7.70×10^5	8.84×10^5	9.90×10^5
16	4.19×10^6	5.22×10^6	6.00×10^6	6.88×10^6
24	1.46×10^7	1.81×10^7	2.07×10^7	2.36×10^7
32	3.35×10^7	4.15×10^7	4.77×10^7	5.44×10^7
40	6.40×10^7	7.96×10^7	9.15×10^7	1.05×10^8
48	1.09×10^8	1.36×10^8	1.56×10^8	1.79×10^8
56	1.71×10^8	2.14×10^8	2.46×10^8	2.82×10^8
64	2.53×10^8	3.16×10^8	3.64×10^8	4.17×10^8

* XORは3Δ，HAとFAは和出力が3Δ，桁上出力が2Δとなる．

さいほどよいので，この結果から引放し法の配列が引戻し法より，また，2の補数表示の配列が絶対値表示のものよりも優れていることがわかる．

7. むすび

2の補数表示の除算について，引放し法，引戻し法の原理，商および剰余の補正を導き，キャリーセーブ方式に基づく除算配列の構成を示した．そしてゲート数，遅延時間およびその積について評価を行い，引放し法の配列が引戻し法より，また2の補数表示の配列が絶対値表示のものよりも優れていることを明らかにした．

参考文献

1) たとえば Brent, R.P. and Kung, H.T.: The Area-Time Complexity of Binary Multiplication, *JACM*, Vol. 28, No. 3, pp. 521-534 (1981), など．
2) 萩原　宏：電子計算機2，演算・制御装置，p. 208，朝倉書店，東京 (1971)．
3) Cappa, M. and Hamacher, V.C.: An Augmented Iterative Array for High-Speed Binary Division, *IEEE, Trans. Comput.*, Vol. C-22, No. 2, pp. 172-175 (1973).
4) 久津輪，文，細川：引放し法による2の補数表示除算配列，昭58信学会情報・システム全大，p. 530 (1983)．
5) Agrawal, D.P.: High-Speed Arithmetic Arrays, *IEEE, Trans. Comput.*, Vol. C-28, No. 3, pp. 215-224 (1979).
6) 堀越　彌訳：コンピュータの高速演算方式，p. 421，近代科学社，東京 (1980)．

(昭和58年11月17日受付)
(昭和59年5月15日採録)