大型計算機の除算方式†

西　本　哲　則††

Abstract

The method of divisisn used in the arithmetic unit of the large scale computer of national project is discussed. Two carry save adders are used during iterations, instead of a carry propagate adder.

Two bits of partial quotients are decided in every iteration from higher bits of the outputs of carry save adders and higher 4 bits of the divisor.

1. 緒　言

この論文は，大型プロジェクトの大型電子計算機に使用した除算方式について述べる．その基本となる考え方は，除算の繰り返しを通常のキャリーを伝播させる加算器を使用せず Carry Save Adder[1] を使用して1回の繰り返しで2ビット商を立てて高速化をはかったものである．

2. 他の電子計算機の除算方式

（1） ノン・レストアリング方式

被除数 (dividend) を x, 除数 (divison) を d とし部分剰余 (partial remainder) を R とすると次の手順を実行していくものである．簡単のため $d>0$ として話をする．

ステップ 1　x を R にセットする．次のステップ2を実行する．

ステップ 2　R の符号が負であったら R に d を加える．R が正であったら d を引く．その後 R を2倍する．ステップ2を必要な桁数だけ実行する．

この方式の欠点は符号をみるために一回のくりかえしで桁上伝播加算器を一回通さなければいけない．桁上伝播加算器を通すと下の桁から一番上の桁まで桁上が伝播するのに時間がかかり演算時間が遅くなる．

（2） HITAC 5020 E/F の除算方式[2]

一度に $R-3d$, $R-d$, $R+d$, $R+3d$ を発生する回路と $R-2d$, $R+2d$ の符号を調べる回路をもっていて，$R-2d$, $R+2d$ の符号によって $R-3d$, $R-d$, $R+d$, $R+3d$ のどれかを選択するものである．

この方式では一回のくりかえしで 2 bit 以上の商を立てることができる．しかしながらこの方式では $3d$ を発生する回路と $R-3d$, $R-d$ または $R+d$, $R+3d$ を発生する2個の桁上伝播加算器と $R-2d$, $R+2d$ の符号を調べる回路が必要となり並列型の電子計算機では部品の量が多くなって適当でない．

しかし直列型の電子計算機では加算器のゲート数はたいしたことはないので適すると思われる．

（3） 正規化方式

剰余と被除数から適当な商をたててその結果を左シフトして正規化する．これをくりかえしていくものである．この場合，シフトする桁数が一定しないので制御が複雑になる．IBM 社の Stretch[3], FACOM 230-60[4] はこの方式を採用している．

（4） 乗算をくりかえして商を求める方式

除数を D, 被除数を N とする．各々のくりかえしで分母と分子に R_k をかけていって分母を1に収束させると分子が商となる．

$$\frac{N}{D}\times\frac{R_0}{R_0}\times\frac{R_1}{R_1}\times\frac{R_2}{R_2}\times\cdots\cdots\times\frac{R_N}{R_N}$$

$$\Longrightarrow NR_0R_1\cdots\cdots R_N=\text{商}$$

$$DR_0R_1R_2\cdots\cdots R_N\Longrightarrow 1$$

R_k の選択が最も重要な点であって次のようにして決定する．除数は次のようにあらわされる．

$$D=1-x$$

D が正規化されているとすると $x\leq 1/2$ になる．もしも R_0 を $1+x$ に等しくとり分母を R_0 倍すると

$$D_1=DR_0=(1-x)(1+x)=1-x^2$$

† Method of Division in a Large Scale Computer, by Tetsunori Nishimoto (7th. dept. Central Research Laboratory Hitachi Ltd.)

†† 日立製作所中央研究所第7部

ここで $x \leq 1/4$ であるので，新しい分母 $DR=D_1$ は次のようになっている．0.11×××．

次に $R_1=1+x^2$ とえらべば

$D_2=D_1R_1=(1-x^2)=1-x^4=0.1111\times\times\times\times$

ここに $x^4 \leq 1/16$ が成立する．

つまり各回のくりかえしでかける数は分母 D_k の2の補数

$R_{k+1}=2-D_k=2-(1-x_k)=1+x_k$

ただし，$x_k=1-D_k$

をとればよいことになる．したがって，除算は次のようにして実行すればよい．

ステップ 1：除数を正規化する．

ステップ 2：テーブル・ルックアプによって最初の乗数 R_0 をきめる．

ステップ 3：D に R_0 をかけて D_1 を求める．

ステップ 4：N に R_0 をかけて N_1 を求める．

ステップ 5：D_k の補数をとって R_k をつくる．

ステップ 6：D_k に R_k をかけて D_{k+1} をつくる．

ステップ 7：N_k に R_k をかけて N_{k+1} をつくる．

ステップ 8：D_k が1になるまで，ステップ 5，6，7 をくりかえす．そして N_k が商に近づいていく．

この方式は高速であるが，剰余が求まらないことと商の最後の1ビットが，下位機種と異なることが難点である．IBM 360-91[1] がこの方式を採用している．

3. 一般的な除算のアルゴリズム

一般に除算は次のようなステップをふんで実行される[5]．被除数を x，除数を d，r を radix，y' を前のくりかえしの部分剰余，y を現在の部分剰余とする q を商とすると

ステップ 1　$y' \leftarrow x$，$q \leftarrow 0$，$k \leftarrow 0$

ステップ 2　$k \leftarrow k+1$，y' と d からある規則にしたがって j を決定する．

ステップ 3　$y \leftarrow y'-jd$

ステップ 4　$y' \leftarrow ry$

ステップ 5　$q \leftarrow rq+j$

ステップ 6　k の値によって繰り返しの回数の終りを判定する．終りでなければステップ2に移る．

n 回繰り返したときの y，y'，q の値をサフィクスをつけて y_n，y'_n，q_n と表わす．

次の関係が成立する．

$$r^{n-1}x=q_n \cdot d+\frac{1}{r}y'_n. \qquad (1)$$

実際 $n=1$ のときにはあきらかに式(1)は成立している．$n=m$ のときに式が成立しているとすると

$$r^{m-1}x=q_m \cdot d+\frac{1}{r}y'_m.$$

r を両辺にかけて

$$r^m x=r \cdot q_m d+y'_m$$
$$=rq_m d+y_{m+1}+jd$$
$$=(rq_m+j)d+\frac{1}{r}y'_{m+1}.$$

したがって

$$r^m x=q_{m+1}d+\frac{1}{r}y'_{m+1}$$

となり $n=m+1$ のときにも式(1)は成立する．

ステップ2での j の決定のしかたによっていろいろな除算の変形ができる．たとえば，ノンレストアリング法では $y'/d \geqq 0$ ならば $j=\frac{1}{2}$，$\frac{q'}{d}<0$ ならば $j=-\frac{1}{2}$ とするものである．このように通常のノンレストアリング法で j を決定した時 $r=2$ の時には $\left|\frac{x}{d}\right|<1$ であれば $\left|\frac{y'}{d}\right|<1$ が保障されて式(1)より n 回の繰り返し後には

$$x=q_n \cdot 2^{-(n-1)}d+y'_n \cdot 2^{-n}$$

が成立し

商 $q_n \cdot 2^{-(n-1)}$　　余り $y'_n \cdot 2^{-n}$

が求められることになる．

4. 大形プロジェクトの除算方式

4.1 構 成

Fig. 1 に全体のブロック・ダイヤグラムを示す．説明の都合上一語の長さを 56 ビットとし各々のレジスターの一番左のビットから 0, 1, 2, 3, ⋯, 55 とする．0ビットが符号ビットで 55 が最下位のビット，小数点の位置は0ビットと1ビットの中間にあるものとする．負の数は2の補数であらわす．

Carry Save Adder の各々の桁は Fig. 2 で示されるように3つの入力があり出力として Carry と Sum を別々に出すものである．Fig. 3 の詳細な Block Diagram からわかるように，上の方の Carry Save Adder の出力は1ビット左方にシフトされて下の方の Carry Save Adder の入力となる．下の方の Carry Save Adder の出力は1ビット左方にシフトされて E，および F レジスタにセットされる．E，F レジスターの上位6ビットのみは Fig. 1　で点線で示されるよ

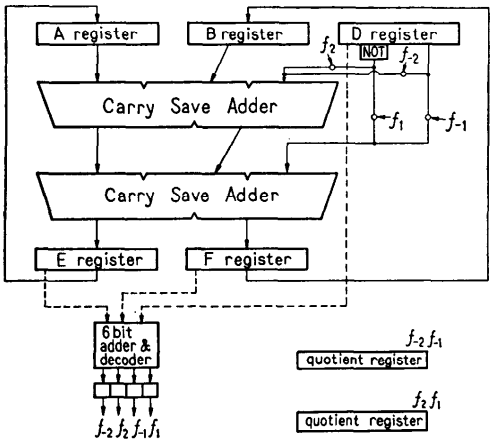

Fig. 1 Block diagram of division

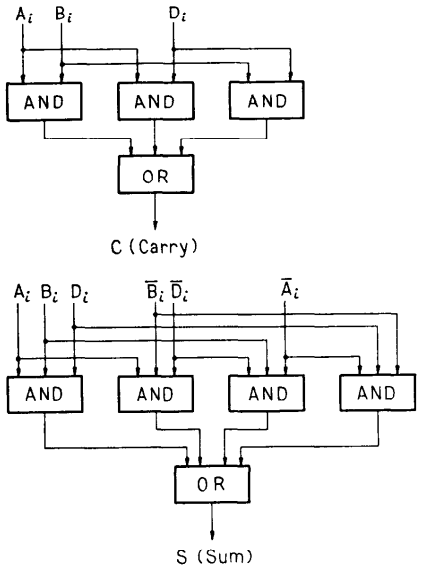

Fig. 2 Carry save adder

うに，加算されてその結果とDレジスタの 2, 3 ビット目により4個の信号 f_2, f_1, f_{-2}, f_{-1} がつくられる．そしてその4個の出力 f_2, f_1, f_{-2}, f_{-1} によって次の繰り返しで Carry Save Adder に加えられる内容が制御される．すなわち f_2 が1であればDレジスター（このレジスターに除数がセットされている）の補数が，また f_{-2} が1であればDレジスターの内容がそのまま上の方の Carry Save Adder に加えられる．また f_1 が1であればDレジスターの補数が f_{-1} が1であればDレジスタの内容がそのまま下

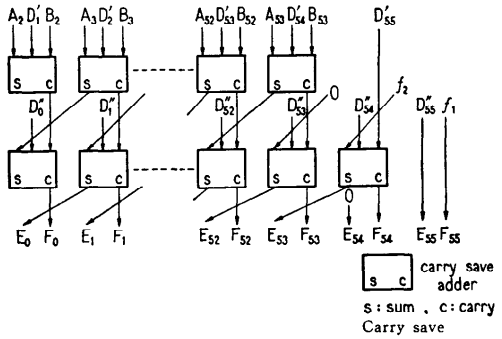

Fig. 3 Detailed diagram of adder tree

の方の Carry Save Adder に加えられる．

なお，補数は2の補数としているからDレジスターの内容を反転（すなわち 0→1, 1→0）しただけでなく最後のビットに1を加える必要がある．それはFig. 3 で示されるように Carry Save Adder の右端があいているためにそこに入力するようにすればよい．

4.2 アルゴリズム

除算は次のようなシーケンスで実行される．

ステップ 1 除数を正規化して被除数，除数を正にする．

ステップ 2 被除数をFレジスタにセットしEレジスタをクリヤーする．Dレジスタには除数をセットする．quotient register 1, 2 をクリヤーする．

ステップ 3 Fレジスタ，Eレジスタの上6ビットを加算しその上位5ビットとDレジスタの 2, 3 ビット目により（Table 1 にしたがって）f_2, f_1, f_{-1}, f_{-2} をつくりだしフリップ・フロップにセットする．Eレジスタ，Fレジスタの内容をそれぞれAレジスタ，Bレジスタに転送する．

ステップ 4 f_2, f_1, f_{-1}, f_{-2} フリップ・フロップの内容にしたがって，Aレジスタ，Bレジスタの内容にDレジスタの内容を f_2, f_1, f_{-1}, f_{-2} により $\pm\frac{3}{4}$, $\pm\frac{2}{4}$, $\pm\frac{1}{4}$, 0倍して2個の Carry Save Adder で加えて2ビット左シフトしてEレジスタ，Fレジスタにセットする．

同時に quotient register 1 の内容は2ビット左シフトされてその右方に f_2, f_1 がつめられる．同様に quotient register 2 も 2 ビット左シフトされて右方

Tablee 1　Decode table.

v \ d	0100-	0101-	0110-	0111-	f_2, f_{-2}, f_1, f_{-1}
011—	$j=\frac{3}{4}$				$f_2=0$ $f_1=1$ $f_{-2}=0$ $f_{-1}=0$
0101-					
01001					
01000				$j=\frac{2}{4}$	$f_2=1$ $f_1=0$ $f_{-2}=0$ $f_{-1}=0$
01000					
00111					
00110					
00101					
00100				$j=\frac{1}{4}$	$f_2=0$ $f_1=1$ $f_{-2}=0$ $f_{-1}=0$
00011					
00010					
00001					
00000			$j=0$		$f_2=0$ $f_1=0$ $f_{-2}=0$ $f_{-1}=0$
11111					
11110					
11101				$j=-\frac{1}{4}$	$f_2=0$ $f_1=0$ $f_{-2}=0$ $f_{-1}=1$
11100					
11011					
11010					
11001				$j=-\frac{2}{4}$	$f_2=0$ $f_1=0$ $f_{-2}=1$ $f_{-1}=0$
11000					
10111					
10110	$j=-\frac{3}{4}$				$f_2=0$ $f_1=0$ $f_{-2}=1$ $f_{-1}=1$
100—					

に f_{-2}, f_{-1} がつめられる．ステップ4が28回繰り返されたかを調べて，もしそうでなければステップ3に進む．

ステップ 5　quotient register 1 から quotient register 2 の内容を通常の加算器で引いて商を求める．EレジスタとFレジスタを加えて余りを求める．後処理を実行する．

4.3　部分商の決定方法

Fig. 1 の f_2, f_1, f_{-2}, f_{-1} をどのようにして決定したらよいか，いいかえると Table 1 にしたがって j を決定していけばよいという保証を以下に述べる．

3 で述べたように

$$y_{k+1}=y'_k-jd,\quad y'_k=ry_k$$

だから　$z_k=\frac{y_k}{d}$,　$z'_k=\frac{y'_k}{d}$　とおいて上式を d で割ると

$$z_{k+1}=z'_k-j,\quad z'_k=r\cdot z_k$$

となる．

Fig. 4 は4進数を用いた場合，部分剰余 y'_k と立てるべき部分商 j との関係を示す図である．横軸を z'_k，縦軸を z_{k+1} とすると，$|z'_k|<1$，$|z_{k+1}|<\frac{1}{4}$ となり図示の長方形の中に，z'_k, z_{k+1} の pair は除算実行中にあるようにしなければいけない．立てられる部分商として $j=-\frac{3}{4}$, $-\frac{2}{4}$, $-\frac{1}{4}$, 0, $\frac{1}{4}$, $\frac{2}{4}$, $\frac{3}{4}$ の7個をえらび，これが Fig. 4 に実線で示してある．除算のくりかえしの途中である z'_k に対して選択される j は z'_k を z 軸に平行に延長したときに点線のなかで交る実線に対応する j でなければならない．もしそうでないとするとそのときの z_{k+1} はその実線との交点の z 座標に等しくその絶対値が$\frac{1}{4}$をこえるからである．従って，ある z'_k が与えられたとき Fig. 4 の枠内において，z_{k+1} 方向に延した線が交叉する直線 $z_{k+1}=z'_k-j$ の j を部分商として選択すれば，次の $z'_{k+1}=4\cdot z_{k+1}$ も $|z'_{k+1}|<1$ が保証され，除算が実行できる．

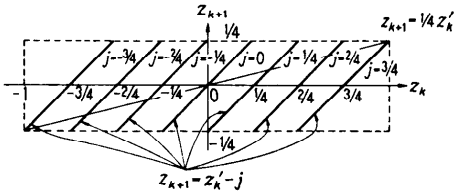

Fig. 4　Relation between z_{k+1} and z'_k

以上から z' に対して選択可能な j がわかったが次に y', d から j をどのように選択すればよいかを示す．

Fig. 5 において横軸は除数 d を，縦軸は部分剰余 y'_k を示しており，原点から放射状に延びている直線 $z'_k=1$, $\frac{3}{4}$, $\frac{2}{4}$, …, 0, …$-\frac{3}{4}$, -1 は，Fig. 4 から明らかなように部分商 j を選択する際の z'_k の分割点になる値である．従って，直線 $z'_k=\frac{3}{4}$ と $z'_k=\frac{2}{4}$ $z'_k=\frac{2}{4}$ と $z'_k=\frac{1}{4}$, …, $z'_k=-\frac{2}{4}$ と $z'_k=-\frac{3}{4}$ の各々に挟まれた領域内に，それぞれ $j=\frac{3}{4}$ と $j=\frac{2}{4}$, $j=\frac{2}{4}$ と $j=\frac{1}{4}$, …, $j=-\frac{2}{4}$ と $j=-\frac{3}{4}$ の各々のいずれかを選択するための境界線を設ける必要がある．

もしも，Carry Save Adder でなく通常の加算器をもちいるのであれば Fig. 5 の実線のあいだに j を定するための境界線をもうければよいのであるが，Carry Save Adder を用いるためにさらに次のようにせばめられる．

Fig. 1 の f_2, f_1, f_{-1}, f_{-2} を作るためにEレジスタおよびFレジスタの上位6ビット（0〜5 ビット）をとりだし両者の和を求め上位5ビット（0〜4 ビット）を取りだしている．この上位5ビットすなわち部分剰余 y'_k の概数を v とする．

v は Fig. 5 に示すように 1/16 間隔の飛びの値を示す．また v と y'_k との誤差を ε とする．

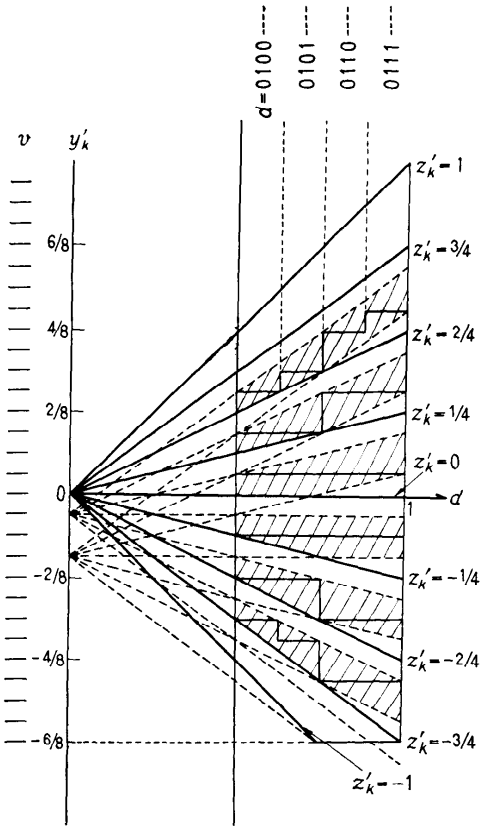

Fig. 5 Decision of j

$$y'_k = v + \varepsilon. \qquad (2)$$

そしてεは，E，Fレジスタの6ビット目以下を切りすてたためにおこる誤差とその和の5ビット目以下を切りすてたためにおこる誤差の和となるから．

$$0 \leq \varepsilon < 2\times 2^{-5} + 2^{-4} < 2^{-3}. \qquad (3)$$

となり次の関係が生ずる．

$$v \leq y'_k < v + 2^{-3}. \qquad (4)$$

式(4)から $-1 < y'_k < -\frac{7}{8}$ の範囲内に部分剰余がくると，vが-1以下の数値になる．これを2の補数であらわしたときには 110.1110 のようになりサインビットより上位のビットは無視されて 0.1110 のように正の1に近い数となって不都合である．

そこで $y'_k = 4 \cdot y_k$ の関係から $-\frac{1}{4} < y_k < -\frac{7}{8}\times\frac{1}{4} < -\frac{3}{16}$ $(k=1,2,3,\cdots)$ にならないように部分商 j を立てる必要がある．y_{k+1} は Fig. 5 において，y'_k とdによって決められる点から，$z'_k = \frac{3}{4}, \frac{2}{4}, \cdots, -\frac{3}{4}$ のいずれかに平行に引いた直線が縦軸と交る点の座標で与えられるから，逆に，縦軸上の座標 $-\frac{3}{16}$ の点から直線 $z'_k = \frac{3}{4}, \frac{2}{4}, \cdots, -\frac{3}{4}$ に平行に引いた点線より下の範囲で，それぞれ，部分商 $j = \frac{3}{4}, \frac{2}{4}, \cdots, -\frac{3}{4}$ を立てることのないようにしなければならない．

また，(4)式から概数vの最後のビット (1/16に相当) は，実際の部分剰余 y'_k と異なっている恐れがあるため，vに 1/16 を加えたものがすぐ上にある直線を越える場合には，その越えた位置に相当する j が立てられるように，前に述べた境界線を設ける必要がある．従って，境界線は Fig. 5 に示されたおのおのの点線より下に設ける必要がある．以上の理由で部分商 jを選択するための境界線は，それぞれ，$z'_k = \frac{3}{4}$ と $z'_k = \frac{2}{4}$，$z'_k = \frac{2}{4}$ と $z'_k = \frac{1}{4}, \cdots, z'_k = -\frac{2}{4}$ と $z'_k = -\frac{3}{4}$にかこまれた領域で，かつ，点線にかこまれた，斜線をほどこされた範囲内に設ける必要がある．斜線の範囲内に境界線を設ける際に，Fig. 5 のように，除数dの$\frac{1}{2} < d < 1$ なる範囲内で，除数 d に無関係に境界線を設けることはできない．そこで，概数vとともに，除数dの上位ビットをも見て，たとえばFig. 5 のごとき階段状の境界線を設けることができる．境界線では除数dの符号ビットを含めて上位4ビットを見た一例を示したが，斜線の範囲内ならば他のどんな境界線を設けても良い．Fig. 5 で実線で示されるように境界をもうけるとしたときの Table が Table 1に示される．

5. 乗除算ユニットへの適用

大型プロジェクトの電子計算機の中央処理装置のなかで乗除算を実行する multiply divide unit は Fig. 6 のようになっている．この multiply divide unit で浮動小数点，固定小数点のすべての乗除算を実行する．4個の Carry Save Adder が使用されており，最初のものと3番目のものは情報を保持する機能をそなえた Carry Save Adder である．

このうち除算に使われるものは 3 番目と 4 番目の Carry Save Adder である．そしてこの Multiply divide unit は乗算のために3倍回路をそなえているため，2つの Carry Save Adder に divisor かその

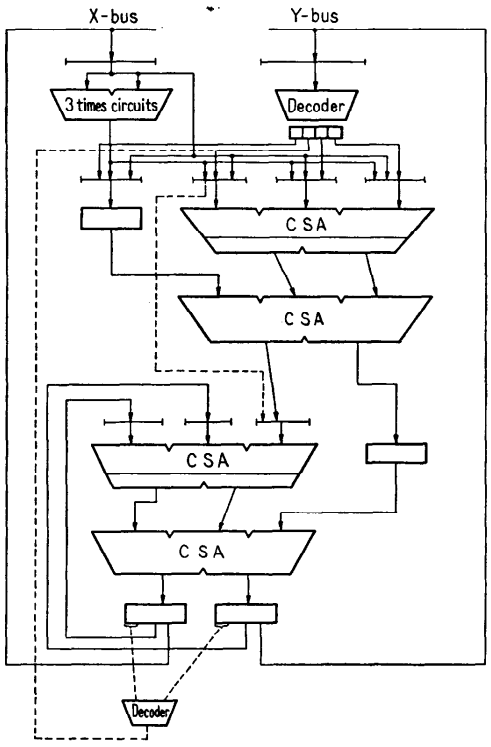

Fig. 5 Block diagram of multiply devide unit

補数をとったものを加えるかわりに，上の方の Carry Save Adder に $\pm\frac{3}{4}d$, $\pm\frac{2}{4}d$, $\pm\frac{1}{4}d$, 0 のどれかを加えることとし1番下の Carry Save Adder は除算のときは素通し（ただし補数をとったときの最後のビットの補正のため，最右端のものだけは使用する）ている．また乗算と，除算で各繰り返しでシフトする方向が逆になるがそれは Fig. 6 の1番下の2個のレジすたから3番目の Carry Save Adder へいくパスを除算用のものをもうけることで解決している．

6. 結 言

大型プロジェクトの電子計算機で使用した除算方式とそれを乗算器にどのように組込んだかを述べた．

なお繰り返しにはいる前の正規化と最後の補正についての問題があるがこれは別に報告する予定である．

謝 辞

ご指導をいただいた当社神奈川工場超大型部中沢部長および石原主任技師に謝意を表わします．また本研究の一部は通産省大型プロジェクトの一環としておこなわれたものでその点に関して関係各位に感謝いたします．

参考文献

1) S. F. Anderson: The IBM System 360 model 91: Floating Execution Uuit, IBM Journal of R. D. January 1962.
2) K. Murata, K. Nakazwa: Very High Speed and Serial-Parallel Computers HITAC 5020 and 5020 E, FJCC 1964 Vol. 26, pp. 187～204.
3) Buchholz, Planning a computer system Project stretch, 1962, McGraw.
4) 林：FACOM 230-60 演算方式について，電気四学会連合大会予稿，昭和44年，3165.
5) J. E. Robertson: A new class of digital division method, Trans, of IRE EC-7, No. 3 Sept. 1958.
6) 中沢，石原：超大型機の演算制御，情報処理，Vol. 12, No. 8 (1971—8) pp. 476～481.
7) 西本：超高速電子計算機の2進並列除算方式，電子通信学会創立50周年記念全国大会予稿，昭和42年，982.
8) T. Nishimoto: U. S. Patent 3, 621, 218.
9) 西本：特許公告番号 47-23214.

(昭和47年9月1日 受 付)
(昭和47年9月28日 再受付)