論文 / 著書情報
Article / Book Information

論題(和文)	実積和演算にも適した複素乗算ユニットの一構成法
Title(English)	A Proposal of Complex Multiplier Suitable for the 4operands Real-Multiply Accumulation
著者(和文)	根岸良征, 渡部英二, 西原明法, 柳澤 健
Authors(English)	Eiji Watanabe, AKINORI NISHIHARA
出典(和文)	電子情報通信学会論文誌, Vol. J84-A, No. 3, pp. 278-286
Citation(English)	, Vol. J84-A, No. 3, pp. 278-286
発行日 / Pub. date	2001, 3
URL	http://search.ieice.org/
権利情報 / Copyright	本著作物の著作権は電子情報通信学会に帰属します。 Copyright (c) 2001 Institute of Electronics, Information and Communication Engineers.

論　文

実積和演算にも適した複素乗算ユニットの一構成法

根岸　良征[†]　　渡部　英二[†]　　西原　明法[††]　　柳澤　健[†]

A Proposal of Complex Multiplier Suitable for the 4operands Real-Multiply Acuumuration

Yoshimasa NEGISHI[†], Eiji WATANABE[†], Akinori NISHIHARA[††], and Takeshi YANAGISAWA[†]

あらまし　マルチメディア分野での応用において，複素信号処理と実信号処理が同時に行われることが多くなり，複素信号処理にも実信号処理にも適した構成の信号処理プロセッサが要求されている．複素演算用の演算ユニットとしては，けた上げ伝搬のない回路を構成でき高速な複素演算回路が実現できるなどの特徴から，冗長数系に基づくものがこれまでに提案されている．また，複素乗算を用いて実演算を行う手法も提案されており，これを用いると，複素演算ユニットで効率的な実演算が可能となる．ところが，従来提案されてきた冗長数系に基づく複素演算ユニットでは，実部けたと虚部けたにおいて重みが異なるなどの理由で実演算処理においては必ずしも最大の性能を発揮できる構成をしていなかった．本論文では，実部けたと虚部けたとで共通の重みをもつ冗長複素数表現と，この表現に基づく部分積生成回路を導入し，高速な実演算処理も可能な複素演算ユニットを示す．

キーワード　複素演算ユニット，複素数表現，積和演算，部分積生成

1. まえがき

近年，ディジタルカメラを内蔵して画像を送信する携帯電話や携帯情報端末に代表されるようなマルチメディア携帯端末が注目されている．マルチメディア携帯端末では，画像処理，音声処理，通信処理などの多岐にわたる高速なディジタル信号処理を必要とするため，信号処理プロセッサを用いてディジタル信号処理を実現することが一般的である [1]．なかでも，携帯情報端末のように，主に実信号処理を主体とする画像/音声のフィルタ処理と，主に複素信号処理を主体とするデータ通信用変復調処理とが混在する事例には，複素数を直接取り扱える複素信号処理プロセッサが適している [2], [3]．特に，文献 [3] で提案した 4 オペランド実積和演算は，複素乗算を用いて実積和演算を効率的に処理できる手法であり，この演算モードを備えた複素信号処理プロセッサは，実演算と複素演算が混在し，高速処理が要求される用途に向いている．

複素信号処理プロセッサの構成上重要な箇所としては，複素乗算ユニットの構成があげられる．複素乗算は，複素数の実部・虚部間で値の移動が発生するために，実演算に比べてより多くの処理を必要とし，高速な演算の実現が困難となるからである．つまり，高速な複素信号処理プロセッサの実現には，高速な複素乗算ユニットの実現が不可欠であると言える．このような背景をもとに，これまでに，いくつかの複素乗算ユニットの構成が提案されている．これまでに提案されてきた複素乗算ユニットには，実乗算器を複素乗算用に拡張したもの [3]，冗長数系を利用した実乗算器を用いたもの [4], [5]，冗長複素数系を利用したもの [6] がある．文献 [4], [5] に示される乗算器は，冗長数系に基づく加算器を用いることで高速な乗算器を実現していることが特徴である．冗長数系は符号付き数系の一種であり，けた上げ伝搬のない加算が行えるので，回路構成時にけた上げ伝搬による遅延を抑制でき，高速処理向きの回路を構成できる．文献 [6] では，冗長数系を更に発展させた冗長複素数系（RCNS）を導入している．冗長複素数系では，複素数を実部虚部の区別のない，完全な一つの数として扱えるために，信号処理プログラムの作成を容易にできる．しかしながら，文

[†] 芝浦工業大学システム工学部，大宮市
Department of Systems Engineering, Shibaura Institute of Technology, 307 Fukasaku, Omiya-shi, 330–8570 Japan
[††] 東京工業大学教育工学開発センター，東京都
Center for Research and Development of Educational Technology, Tokyo Institute of Technology, nn2-12-1 O-okayama, Meguro-ku, Tokyo, 152–8552 Japan

献 [5], [6] に示される乗算器では，4 オペランド実積和演算は行えず，更に，文献 [5] で提案されている複素乗算器では，複素数の実部と虚部を分離して扱うことを前提としているため，1 複素数を 1 オペランドとして扱う方法が考慮されていない．複素数を実部オペランド，虚部オペランドとして 2 オペランドで扱うと，4 オペランド実積和演算時に八つのオペランドを操作することになるので，プログラムの作成を困難にしており，複素数を 1 オペランドで扱う手法が望まれている．

一方で，4 オペランド実積和演算が可能な文献 [3] の乗算器は，通常の数系に基づく実数乗算器で構成しているため，けた上げ伝搬が障害となって部分積加算部が高速に動作しづらく，高速に動作させるために，乗算器内部を 4 段のパイプラインで構成している．4 オペランド実積和演算はパイプラインハザードによる処理の中断に敏感で，実演算・複素演算時と比べて 2 倍から 4 倍の復旧コストが必要である．冗長表現を用いた加算器は通常の数系に基づく加算器よりも高速に動作可能であるから，冗長表現を用いた加算器を用いることにより，通常の数系に基づく部分積加算部よりも高速な部分積加算部を構成でき，パイプラインの段数を節減する効果が得られる．

そこで，本論文では，冗長複素数系の特徴を用いた乗算回路によって高速な複素乗算を実現しつつ，1 複素数を 1 オペランドで扱え，4 オペランド実積和演算を効率的に処理できる複素乗算ユニットを提案する．そのために，まず，冗長複素数系を用いた従来形の乗算ユニットにおいて 4 オペランド実積和演算をそのまま適用した場合について述べ，出力される結果の実部けたと虚部けた間の演算結果の重みが異なることが原因となって効率的に処理を進められないという問題点を明らかにする．次に，この問題点を解決するために，新しい複素数の表現方法を導入する．この表現方法を用いると，実部けたと虚部けたの重みが等しくなるので，4 オペランド実積和演算の結果，二つの重みも等しくなり，4 オペランド実積和演算を使った処理を迅速に進められる．最後に，導入する複素数の表現法において，高速に部分積を生成するためのアルゴリズムと，このアルゴリズムを実現する回路構成について述べる．

2. 複素乗算を用いた 4 オペランド実積和演算

二つの複素数 $(a+bj)$，$(c+dj)$ の乗算は，

$$(a+bj)\cdot(c+dj)=(ac-bd)+(ad+bc)j \quad (1)$$

である．ここで，式 (1) の結果に注目すると，結果の実部は $(ac+(-b)d)$ とみることができ，四つの実数 a，$-b$，c，d の積和演算とみなすことができる．このような見方をした場合，結果の虚部は不要である [3]．

さて，一般的な複素演算ユニットでは，乗算結果の実部を計算する回路と虚部を計算する回路の二つの回路の並列動作によって複素数の乗算を実行している．つまり，虚部の演算が不要であるとすると，この部分の回路が効果的に使われないということになる．そこで，この部分の回路を別の二つの複素数 $(e+fj)$，$(g+hj)$ の乗算，

$$(e+fj)\cdot(g+hj)=(eg-fh)+(eh+fg)j \quad (2)$$

に用いることを考える．今度は，乗算結果の虚部を必要とし，実部を不要とする．すなわち，ここでの演算では四つの複素数 $(a+bj)$，$(c+dj)$，$(e+fj)$，$(g+hj)$ において $\mathrm{Re}[(a+bj)(c+dj)]$ を求めるのに必要な演算と $\mathrm{Im}[(e+fj)(g+hj)]$ を求めるのに必要な演算とを一つの複素演算ユニットを用いて同時に実行することにする．この処理の結果，$(ac+(-b)d)$，$(eh+fg)$ が得られるので，同時に 2 回の実積和演算を行ったと考えられ，結果的に，4 回の乗算と 2 回の加算を 1 回の複素乗算で処理したと見ることができる．本論文では，この演算を 4 オペランド実積和演算と呼ぶ．

3. 4 オペランド実積和演算に適した数表現

これまでに示したように，4 オペランド実積和演算は，複素乗算器の実部処理部と虚部処理部を別々の演算のために用いる．しかしながら，4 オペランド実積和演算は複素乗算を用いることから出力結果は一つの複素数として得られる．従来，複素信号処理プロセッサ上では，複素数は実部と虚部を分離して 2 オペランドとして扱う必要があり，この方法では，4 オペランド実積和演算時に八つのオペランドを操作する必要が生じていた．そこで，1 オペランドで複素数を表せる数表現をもつ数系として冗長複素数系（RCNS）に注目する．しかしながら，RCNS では，実部けたの重みと虚部けたの重みが異なるために，4 オペランド実積和演算時に，この重みの違いが問題となる．例えば，RCNS [6] に基づく表記の複素数 $\mathcal{A}$，$\mathcal{B}$，$\mathcal{C}$，$\mathcal{D}$ に対して，4 オペランド実積和演算を適用する場合を考えると，RCNS では，基数を $2j$ として，

$$[a_m \cdots a_1 a_0 . a_{-1} \cdots a_{-l}] = \sum_{i=-l}^{m} a_i \cdot (2j)^i \qquad (3)$$

と複素数を表記するから，4 オペランド実積和演算の結果は，

$$\begin{aligned} \mathcal{S} &= \mathrm{Re}[\mathcal{A} \cdot \mathcal{B}] + j\mathrm{Im}[\mathcal{C} \cdot \mathcal{D}] \\ &= \sum_{i=-l}^{m} s_i \cdot (2j)^i \end{aligned} \qquad (4)$$

として得られる．ただし，l，m は正の整数，j は虚数単位，$a_i, s_i \in \{3, 2, 1, 0, \bar{1}, \bar{2}, \bar{3}\}$ である．式 (4) において，$l = 0$，$m = 7$ とした場合の図を図 1 に示す．

式 (4) からわかるように，4 オペランド実積和演算の結果は，RCNS に基づく表記の複素数一つとして得られるが，積実部と積虚部が独立した積和演算の結果となっているので，実際には，$\mathrm{Re}[\mathcal{S}]$ と $\mathrm{Im}[\mathcal{S}]$ の二つの演算結果が得られたことになる．積和演算を行う多くの事例では $\mathrm{Re}[\mathcal{S}]$ と $\mathrm{Im}[\mathcal{S}]$ を更に足し合わせる場合が多い．信号処理プロセッサでは，これらの加算は乗算器の次段にある加算器あるいは ALU で行われるが，図 1 の出力の場合，$\mathrm{Re}[\mathcal{S}]$ から得られる数は，けたの重みが $\{1, \bar{4}, 16, \bar{64}\}$ であるのに対して，$\mathrm{Im}[\mathcal{S}]$ の重みは，$\{2, \bar{8}, 32, \bar{128}\}$ であるから，$\mathrm{Re}[\mathcal{S}] + \mathrm{Im}[\mathcal{S}]$ なる加算を行う際に，各けたを直接加算することができず，重み変換を行う必要が生じる．すなわち，この機能を果たす回路を新たに乗算器と加算器の間に付加せねばならず，回路構成を複雑にする要因となる．

そこで，これらの重みが等しくなるような表現として図 2 に示す表現を導入することを提案する．この表現では，

$$\begin{aligned} &[b_{2m+1} \cdots b_1 b_0 . b_{-1} \cdots b_{-2n}] \\ &\quad = \sum_{k=-2n}^{2m+1} \{b_k \cdot 2^k + j(b_{2k+1} \cdot 2^k)\} \end{aligned} \qquad (5)$$

である．ただし m, n は正の整数，$b_k \in \{1, 0, \bar{1}\}$ である．この表現法では，実部に対する重みと虚部に対する重みが等しいため，4 オペランド実積和演算の二つの出力の重みが等しくなり，後の加算において特別な処理を必要としないことが特徴である．この数表現の数系はもはや基数 $2j$ の RCNS ではなく，虚部，実部がそれぞれ基数 2 の数系で，それが交互に配置されている．ただし冗長数系であることには変わりはない．

ここで，トランスバーサル形回路を例に，従来形乗

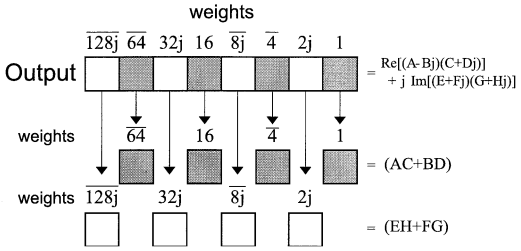

図 1　従来の複素演算装置からの出力
Fig. 1　The conventional output format of the complex arithmetic unit.

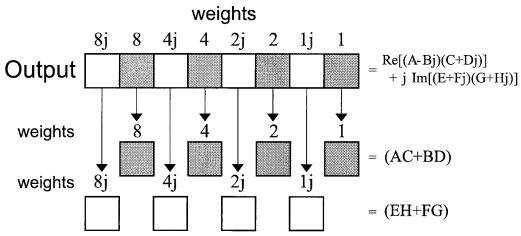

図 2　提案する数表現
Fig. 2　The proposed complex number representation.

算ユニットと提案した数系を採り入れた乗算ユニットにおける 4 オペランド実積和演算手順を考える．トランスバーサル形フィルタは，

$$y(n) = \sum_{i=0}^{M} h_i \, x(n-i) \qquad (6)$$

と表せる．この回路を 4 オペランド実積和演算で処理する場合は，

$$\begin{aligned} y(n) &= \Big\{ h_0 \, x(n) + h_1 \, x(n-1) \Big\} + \cdots \\ &\quad + \Big\{ h_{M-1} \, x(n-M+1) + h_M \, x(n-M) \Big\} \\ &= \Big\{ h_0 \, x(n) - H_1 \, x(n-1) \Big\} + \cdots \\ &\quad + \Big\{ h_{M-1} \, x(n-M+1) + h_M \, x(n-M) \Big\} \end{aligned} \qquad (7)$$

のように，乗算定数 h_i の符号を調節して考える．式 (7) を，従来形数表現を使って処理した場合と，提案した数表現を使って処理した場合の手順を以下に示す．

《従来形数表現による手順》

（1）　$h_0 + j\,(-H_1)$ と $x(n) + jz(1)$，$h_2 + jh_3$ と

$z(3)+jz(2)$ によって 4 オペランド実積和演を行う．（前半の結果を r_1，後半の結果を r_2 とする）

（2） r_2 の重みを r_1 の重みに変換する．

（3） r_1+r_2 を計算する．

《提案する数表現による手順》

（1） $h_0+j\,(-H_1)$ と $x(n)+jz(1)$，h_2+jh_3 と $z(3)+jz(2)$ によって 4 オペランド実積和演を行う．（前半の結果を r_1，後半の結果を r_2 とする）

（2） r_1+r_2 を計算する．

いずれの手順も，4 オペランド演算は複素乗算器で処理し，r_1，r_2 の加算は，乗算器の後段にある ALU で処理することを前提としている．従来形数表現を使って処理する手順の場合では，手順（2）の処理を行うために，実部けた（図 1 の網掛部）と虚部けた（図 1 の白部）の重みを一致させるための回路が必要となる．一方で，提案する数表現を用いると，図 2 に示すように実部けたと虚部けたの重みが一致しているので，すぐに ALU で加算することができる．

次に，この数表現を 2 進数として扱うことを考えてみる．導入した数表現の各けたは $\{1,0,\bar{1}\}$ であるからよく用いられるように $\{1,0,\bar{1}\}$ をそれぞれ $\{01,00,10\}$ と表記することにする．左側のビットを m ビット（マイナスビット），右側のビットを p ビット（プラスビット）と呼ぶ．図 2 に示す 8 けたの数を 2 進数表現すると各けた 2 ビットとなって，全体で 16 ビットである．一方で，図 1 を 2 進数表現すると各けた 4 ビット必要であることより，全体で 32 ビットとなる．

4. 部分積生成部

複素乗算器は部分積生成部と部分積加算部とから成る．前章で導入した数表現は，もはや基数 $2j$ の RCNS ではないので，従来の RCNS 乗算器の部分積生成部は利用できず，部分積生成部を新たに構成する必要がある．部分積の生成法としては拡張 Booth のアルゴリズムによって，加算器を節減する方法が広く知られているが，冗長数系においては，けたに負の値をとるために部分積の種類を減らすことができず，拡張 Booth のアルゴリズムを適用することが困難である．また，ここで導入した複素数表現は実部と虚部が交互に配置されているが，このことも Booth のアルゴリズムを適用することを困難にしている要因の一つである．そこで，本論文では部分積の生成法として，セレクタ方式による部分積生成法とシフト加算方式による部分積生成法を提案する．

4.1 セレクタ式部分積生成

ここでは，例として，実部/虚部ともに 1 けたの複素数の乗算，

$$z=(a+bj)\times(c+dj) \tag{8}$$

を考えてみる．なお，$a,b,c,d\in\{1,0,\bar{1}\}$ である．

この式の計算を行うために，展開した式である

$$z=(a+bj)\times c+(a+bj)\times jd \tag{9}$$

を考えると，式 (9) の前半の項において，部分積 ac は z の実部，部分積 bc は z の虚部にいくことがわかる．ここで a が c によって制御されると考えれば，部分積 ac は c の値によって制御され $\{a,0,\bar{a}\}$ のいずれかの値をとると言える．更に，a を 2 進数表示した (a_m,a_p) の表記より，$\bar{a}$ は a の m ビットと p ビットを入れ換えることにより導き出せることもわかる．これは部分積 bc の算出においても同様のことが言える．したがって，部分積 ac，bc は，乗数 c を制御パラメータとして被乗数の (a_m,a_p)，(b_m,b_p) を制御して決定することができる．この制御規則を表 1 に示す．表 1 は，c が 1 の場合 $(c_mc_p=01)$，部分積 ac は被乗数の実部 $\{a_ma_p\}$ そのものであり，部分積 bc は被乗数の虚部 $\{b_mb_p\}$ そのものであることを示している．また，c が 0 の場合 $(c_mc_p=00)$ は，ac，bc 共に 0，c が $\bar{1}$ $(c_mc_p=10)$ の場合は，ac，bc は，それぞれ被乗数の実部/虚部の p ビットと m ビットを入れ換えたものであることを示している．以上の操作は，(a_m,a_p)，(b_m,b_p) を入力とし，c_m，c_p を制御パラメータとする AND-OR セレクタ回路として図 3 の回路で実現できる．

次に，式 (9) の後半の項について考える．式 (9) の後半の項においては，部分積 ad は z の虚部，部分積 bd は z の実部にいく．この場合も先に行った解釈同様，乗数 d によって被乗数 a，b が制御されると考えられる．つまり，部分積 ad，bd は乗数 d を制御パラメータとして被乗数の (a_ma_p)，(b_mb_p) を制御して決定できる．この制御規則を表 2 に示す．表 1 の場合

表 1　部分積生成規則 I
Table 1　Partial product generation I $((\mathrm{a}+\mathrm{b}j)\times c)$.

multiplier	partial product output	
c_m c_p	real part	imag. part
0 1	a_m a_p	b_m b_p
0 0	0 0	0 0
1 0	a_p a_m	b_p b_m

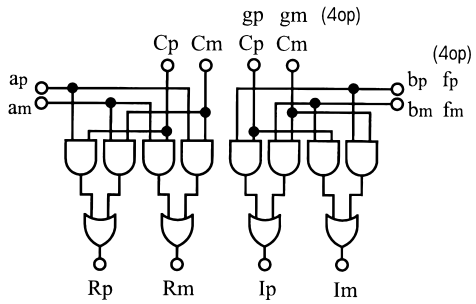

図 3　部分積生成回路 I
Fig. 3　The circuit for the PPG I.

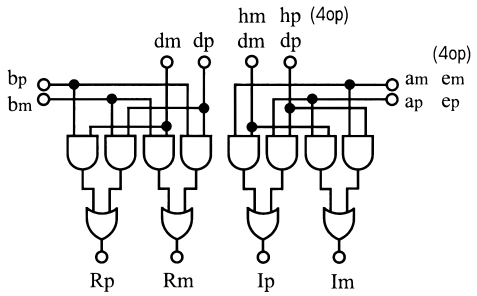

図 4　部分積生成回路 II
Fig. 4　The circuit for the PPG II.

表 2　部分積生成規則 II
Table 2　Partial product generation II $((\mathrm{a}+\mathrm{b}j)\times\mathrm{d}j)$.

multiplier	partial product output	
d_m d_p	real part	imag. part
0 1	b_p b_m	a_m a_p
0 0	0 0	0 0
1 0	b_m b_p	a_p a_m

表 3　シフト加算による部分積生成規則
Table 3　Partial product generation rule for shift-based algorithm.

partial product	multiplier (RB)
$3A$	11
$2A$	10
A	01, $1\bar{1}$
0	00
$-A$	$0\bar{1}$, $\bar{1}1$
$-2A$	$\bar{1}0$
$-3A$	$\bar{1}\bar{1}$

と異なり，被乗数の実部が z の虚部へ，被乗数の虚部が z の実部へ移動する．表 2 の規則は図 4 の回路で実現できる．

以上より，式 (8) の部分積は，セレクタを基本とした図 3，図 4 の回路ですべて求めることができることがわかる．また，**3.** で述べたとおり，4 オペランド実積和演算は，虚部部分積を生成する回路に新たなオペランドを入力するように回路を構成することのみで実現できる．本論文では，この手法による部分積生成法をセレクタ式部分積生成法と呼ぶ．

4.2　シフト加算式部分積生成

次に，乗数を 2 けた単位で扱う方法を考えてみる．ここでは，被乗数の実部/虚部けたが k けた，乗数の実部/虚部けたが 2 けたの複素乗算

$$z = (A + B) \times (C + Dj) \tag{10}$$

を考える．ここで A，B，C，D の各けたは $\{1, 0, \bar{1}\}$ をとる．すると，C，D が取り得る値は，$\{11, 10, 01, 1\bar{1}, 00, 0\bar{1}, \bar{1}1, \bar{1}0, \bar{1}\bar{1}\}$ の 9 通りである．例として，部分積 AC の生成規則を表 3 にあげる．被乗数 A が冗長数系であることを考慮すると，符号の反転は，m ビットと p ビットの交換で実現できるので，$-A$，$-2A$，$-3A$ は A，$2A$，$3A$ の生成回路を利用することができる．したがって，回路構成上は，$-A$，$-2A$，$-3A$ 生成用の回路をあらためて付加する必要はない．また，被乗数の 3 倍を求めるアルゴリズムは，

$$\begin{aligned} 3A &= \sum_{i=0}^{k} a_i \cdot 2^i + \sum_{i=0}^{k} a_{i-1} \cdot 2^i \\ &= \sum_{i=0}^{k} (a_i + a_{i+1}) \cdot 2^i \end{aligned} \tag{11}$$

のように一つ下のけたと加算を行えばよい．ただし，式 (11) において，$a_{-1} = 0$ とし，$i = k$ のときのけた上げは，a_{k+1} に入るものとする．ここでは，$a_i \in \{\bar{1}, 0, 1\}$ の冗長数系を用いているので，式 (11) における加算においてもけた上げ伝搬は生じない．このアルゴリズムを実現する回路を図 5 に示す．

図 5 に示した回路では，式 (11) を用いて被乗数の 0 倍，1 倍，2 倍，3 倍を求めている．被乗数の 1 倍，2 倍は，それぞれ，式 (11) の後半の項，前半の項を 0 にすれば被乗数の 3 倍を求める回路と同一の回路を用いることができる．また，加算器の前段にある符号変換器（Sign Changer）は同一の加算回路で表 3 の正負の演算を実現するためのものである．

以上のように，符号反転が容易であるという冗長数系の特徴を利用しつつ，式 (11) を用いて部分積を生成する方法を，本論文ではシフト加算式部分積生成法

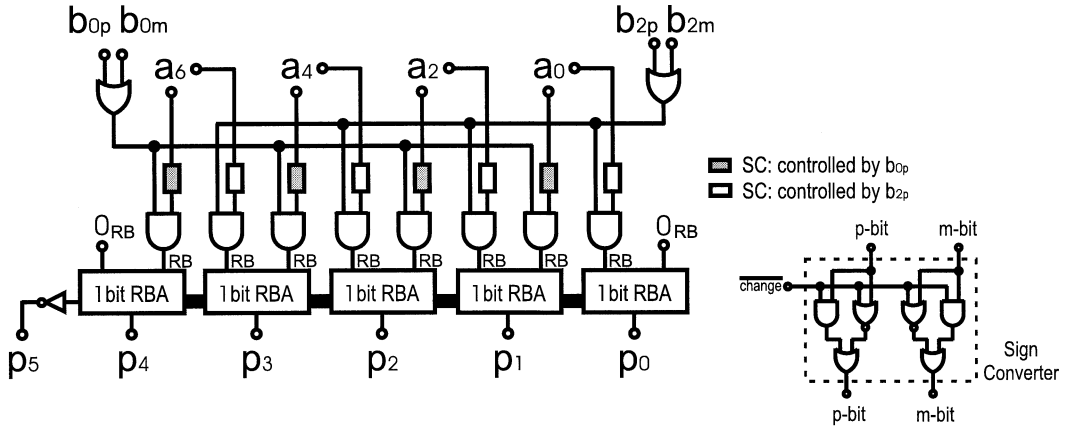

図 5　シフト加算式部分積生成回路（乗数 b_2b_0 の場合）
Fig. 5　Partial Product Generator for the proposed shift-based algorithm.

と呼ぶ．

5. 4 オペランド実積和演算を実現する乗算器構成

前章で述べた，セレクタ式部分積生成法による部分積生成回路を用いた複素乗算器を図 6 に，シフト加算式部分積生成法による部分積生成回路を用いた複素乗算器を図 7 に示す．オペランドは，乗算器の入力部で冗長数系に変換したのち，乗算器に続く ALU の出力部で通常の数系に変換する．

図 6，図 7 に示す乗算器は，図 2 の数表現による 8 けた（実部 4 けた，虚部 4 けた）の複素乗算器であり，実演算，複素演算のほか，4 オペランド実積和演算を実現している．実演算，複素演算，4 オペランド実積和演算は，図 8 中の演算モード信号（MODE）によって切り替わる．複素演算と 4 オペランド実積和演算においては，乗算処理部（部分積生成部，部分積加算部）の動作は全く同じであり，オペランドを取り込むバスが異なる．図 6，図 7 の回路では，いずれも，複素演算時には演算モード信号の制御によって，$\alpha = \gamma$, $\beta = \delta$ として積実部演算部と積虚部演算部に同一のオペランドを与え，4 オペランド実積和演算時には，専用バスからのオペランド γ，δ を積虚部に与えることにより，複素演算も 4 オペランド実積和演算も可能としている．また，実演算は虚部が 0 である複素演算として実行する．

図 6，図 7 の RBA ブロックは，図 9 に示す冗長 2 進加算器で構成されており，それぞれのけた数に応じて並列に接続されている．表 4 は，これらの回路の最長ロジックパスと回路規模，4 オペランド実積和演算への適応を示すものである．表 4 中の RCNS-conv. は，冗長複素数系に基づく従来の数表現で構成した複素乗算器，selector は，セレクタ式部分積生成回路を用いた複素乗算器，shift-based は，シフト加算式部分積生成回路を用いた複素乗算器を表している．また，4op-RMA の欄は，4 オペランド実積和演算への適応を表すもので，RCNS-conv. では，4 オペランド実積和演算の実行は可能ではあるが，重み変換を行う必要があるので，selector，shift-based と比べると適応性が低い，言い換えると，効率的に 4 オペランド実積和演算を実行できないことを表している．

セレクタ式部分積生成法は，基数 2 の部分積生成法，シフト加算式の部分積生成法は基数 4 の部分積生成法と考えることができる．シフト加算式では，図 5 に示す部分積生成部の内部で，被乗数 A の 3 倍の部分積 $3A$ を $2A$ と A の部分積に分解して加算して生成するので，乗算器全体として見た場合，セレクタ式の部分積生成法と比べて加算器の段数に変化がない．しかしながら，シフト加算式では，部分積の生成過程においてシフト演算を利用するために，表 4 に示すように，セレクタ式に比べて回路規模が少なくて済む．したがって，これらの特徴をまとめると，セレクタ式部分積生成法を用いた複素乗算器はシフト加算式部分積生成法を用いたものよりも素子数が多いが，最長ロジックパスが短く高速動作が可能であり，一方，シフト加算式部分積生成法を用いたものは，セレクタ式部

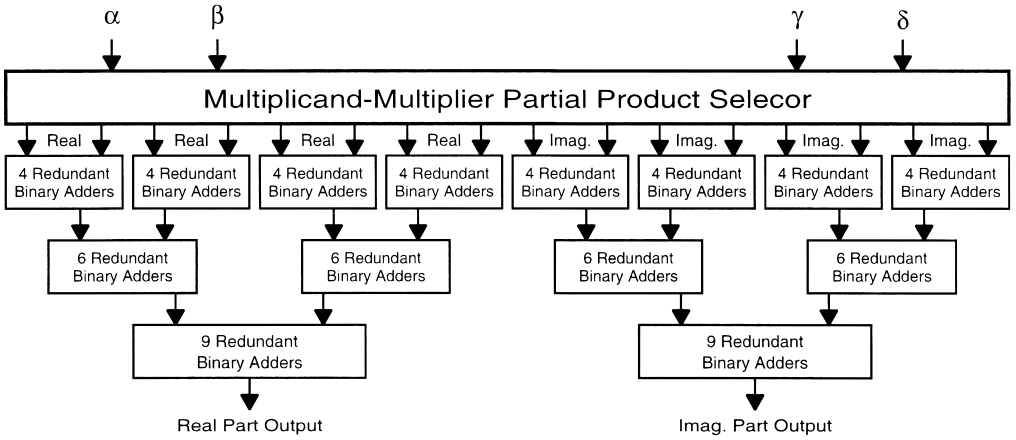

図 6　セレクタ式乗算器の構成
Fig. 6　The proposed complex multipier with Partial Product Selectors.

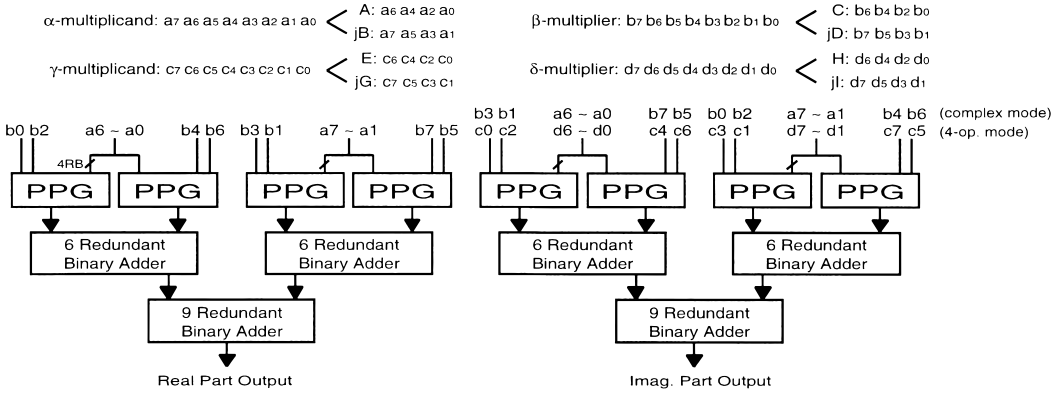

図 7　シフト加算式乗算器の構成
Fig. 7　The proposed complex multiplier with shift-based Partial Product Generators.

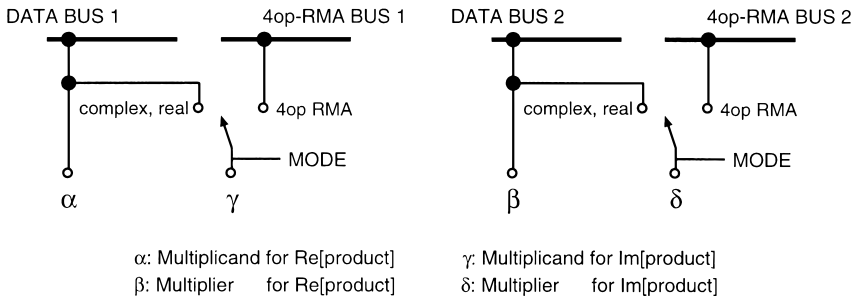

図 8　乗算器演算モードの制御
Fig. 8　Calcuration mode control.

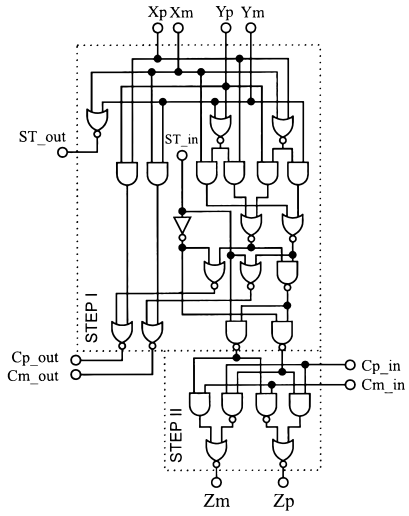

図 9　冗長 2 進加算器
Fig. 9　Redundant Binary Adder.

表 4　従来法との比較
Table 4　A comparison.

	logic path	circuit size	4OP–RMA
RCNS-conv.	26	> 3000	△
selector	23	3976	◎
shift-based	24	2690	◎

分積生成法を用いたものに比べて最長ロジックパスが長いが，素子数が少なくて済む点が特徴であると言える．更に，図 6，図 7 の回路いずれも，RBA ブロックを区切りとしたパイプライン構成にすることが容易に可能である．

6. むすび

本論文では，4 オペランド実積和演算を効率良く処理できる複素乗算器の数表現と構成法を提案した．本論文で提案した数表現は，実部けたと虚部けたを有しており，一つの数で一つの複素数が表現できることや冗長数系であること，虚部けたと実部けたとで重みが等しいので，4 オペランド実積和演算向きであることといった特徴がある．この数表現の導入により，特にトランスバーサル形の回路の処理において高速処理が可能であることを併せて示した．更に，この数表現を使った乗算器を二つ提案した．一つは，セレクタ式部分積生成回路をもつ乗算器で，部分積の生成を高速行えるという特徴がある．もう一つは，シフト加算式部分積生成回路をもつ乗算器で，回路規模が小さくて済むという特徴がある．いずれの回路も，パイプライン構成をとりやすいという特徴があるので，高速動作を行う DSP の乗算器として有用である．今後の課題としては，拡張 Booth's アルゴリズムのように，部分積を正確に半分にする手法の検討や，回路の省電力化設計が考えられる．

謝辞　本研究の一部は，文部省科学研究費補助金 No.10650378 基盤研究 (C)(2) 「システムの複素及び超複素化による高度並列ディジタル信号処理」の補助を受けている．ここに謝意を表する．

文　献

[1] K.J. Rayliu, A. Wu, A. Raghupathy, and J. Chen, "Algorithm-based low-power and high-performance multimedia signal processing," Proc. IEEE, vol.86, no.6, pp.1155–1202, June 1998.

[2] B. Barazesh, J.C. Michalina, and A. Picco, "A VLSI signal processor with complex arithmetic capability," IEEE Trans. Circuits & Syst., vol.35, no.5, pp.495–505, May 1988.

[3] Y. Negishi, E. Watanabe, A. Nishihara, and T. Yanagisawa, "Performance enhancement on digital signal processors with complex arithmetic capability," IEICE Trans. Fundamentals, vol.E82-A, no.2, pp.238–245, Feb. 1999.

[4] Y. Harata, Y. Nakamura, H. Nagase, M. Takigawa, and N. Takagi, "A high-speed multiplier using a redundant binary adder tree," IEEE J. Solid-State Circuits., vol.sc-22, no.1, pp.28–34, Feb. 1987.

[5] K. Shin, B. Song, and K. Bacrania, "A 200-MHz complex number multiplier using redundant binary arithmetic," IEEE J. Solid-State Circuits., vol.33, no.6, pp.904–909, June 1998.

[6] T. Aoki, H. Amada, and T. Higuchi, "Design of real/complex reconfigurable arithmetic circuits using redundant complex number systems," IEICE Trans. Inf. & Syst., vol.J80-D-I, no.8, pp.674–682, Aug. 1997.

[7] Y. Harata, Y. Nakamura, H. Nagase, M. Takigawa, and N. Takagi, "A high-speed multiplier using a redundant binary adder tree," IEEE J. Solid-State Circuits., vol.SC-22, no.1, pp.28–34, Feb. 1987.

[8] Y. Negishi, E. Watanabe, A. Nishihara, and T. Yanagisawa, "A complex multiplier using redundant binary adders with partial product selector," Proceedings of the 1999 ITC-CSCC, vol.I, pp.84–87, Sado, Japan, July 1999.

（平成 11 年 11 月 25 日受付，12 年 7 月 24 日再受付）

根岸　良征　（正員）

平 7 芝浦工大・システム工・電子情報システム卒．平 12 同大大学院博士後期課程了．博士（工学）．VLSI 信号処理についての研究に従事．ACM 会員．

渡部　英二　（正員）

昭 56 電通大・電気通信・電波通信卒．昭 58 同大大学院修士課程了．昭 61 東工大大学院理工・電子物理博士後期課程了．工博．同年同大大学院総合理工・物理情報助手．平 3 芝浦工大・システム工・電子情報システム講師．平 7 同助教授．平 12 同教授，現在に至る．ディジタルフィルタを中心に離散時間回路網の構成と実現の研究に従事．電気学会，IEEE 会員．

西原　明法　（正員）

昭 48 東工大・工・電子物理卒．昭 53 同大大学院博士課程了．工博．同年より同大勤務．現在，同大教育工学開発センター教授．回路理論，電子回路，信号処理等の研究，教育に従事．信号処理用プロセッサのコンパイラ等にも興味をもつ．IEEE, EURASIP, ECS 会員．

柳澤　　健　（正員）

昭 28 東工大・工・電気卒．昭 33 同大大学院博士課程了．工博．同年より同大助手．昭 45 同大教授．平 3 芝浦工大・システム工・電子情報教授，現在に至る．昭 61 本会業績賞受賞．回路理論，電子回路，信号処理等の研究，教育に従事．電気学会会員．