〔原著論文〕

ロボット制御用超高速座標逆変換プロセッサの構成

亀山充隆* 江上秀樹* 樋口龍雄*

ロボットマニピュレータにおいて，手先の位置・姿勢から各関節の変位を，解析解に基づいて計算する，LSI向き超高速座標逆変換プロセッサを提案している．このような逆変換演算を，主に座標の回転移動と逆正接演算の組み合わせにより，効率よく計算する高速アルゴリズムを考案し，これらの演算をCORDIC手法に基づき，高速に処理するハードウェアアルゴリズムについて考察している．これにより，従来にない高速性やコンパクト性，種々のマニピュレータに適用可能であるという汎用性を兼ね備えた専用プロセッサを構成し，ブレッドボードによる動作実験結果を示している．最後に，2 μm CMOS技術に基づくLSI化を前提として，本プロセッサの処理速度やハードウェア量を評価している．処理速度については，通常処理時間が長いとされている6自由度垂直多関節型ロボットにおいて，従来の汎用マイクロプロセッサを用いた場合と比較して大幅な速度向上が可能である．また，ハードウェア量は，8ビットワンチップマイクロコンピュータと同程度の規模であり，LSI化は極めて容易であると考えられる．

1. はじめに

ロボットマニピュレータの制御では，動作経路を作業座標におけるマニピュレータ先端の位置・姿勢で与えるのが一般的であり，これを各関節の変位に変換する座標逆変換の高速化は，極めて重要な問題である．特に高度なロボットの制御においては，作業環境に適応した動作を実時間で実行する必要があり，逆変換を高速に処理しなければならない[1,2]．この逆変換には，1つ1つのマニピュレータに対する解析解を求め，それに基づいて数値計算を行うことにより，関節変位を求める方法や，ヤコビ行列の逆行列を用いて一般的なマニピュレータに対して逆変換を行う方法などがある[3]．本論文では，演算量の観点と共に，従来のほとんどの産業用ロボットでは既に解析解が求められていることを考慮して，前者の解析解が与えられた場合についての逆変換を扱うことにする．この解析解の数値計算は，関数演算数そのものはあまり多くないにもかかわらず，特殊関数演算を数多く含むとともに，各関節に共通の演算が少ないため，処理の高速化が困難であった．一報告例として，7台のCPU，14台のAPU (Arithmetic Processing Unit) によるマルチマイクロプロセッサを用いて，並列処理を行うことにより，この逆変換の処理を3.3msecで実行できることが示されている[4]．本論文では，逆変換演算が，座標の回転移動と逆正接演算の組み合わせで，効率よく計算できることに着目し，これらの演算をCORDIC手法[5]を採用することにより，高速に処理するアルゴリズムについて述べている．さらに，マイクロプログラム制御方式に基づくハードウェアにより構成される，LSI向き専用プロセッサを提案している．最後に，このプロセッサのLSI化を前提とし，処理速度やハードウェア量を評価している．

原稿受付 1987年3月12日

* 東北大学工学部電子工学科

2. 座標逆変換アルゴリズム

2.1 マニピュレータの座標逆変換

現在，さまざまな形態をもつマニピュレータが，産業用あるいは研究用として開発されているが，作業領域で任意の位置・姿勢を与えるためには最低6自由度を持ったマニピュレータが必要となる．ここで，手先の位置・姿勢P，および各関節の変位θを

$$P=(P_1, P_2, P_3 \cdots P_m) \tag{1}$$

$$\theta=(\theta_1, \theta_2, \theta_3 \cdots \theta_n) \tag{2}$$

と書くことにする．mは作業座標の次元，nはアームの自由度を表す．

θからPを求める変換 $P=f(\theta)$ は，順変換と呼ばれ，一般的に求められるが，その逆変換

$$\theta=f^{-1}(P) \tag{3}$$

は，通常複雑な非線形変換であり，一般的な解析解は得

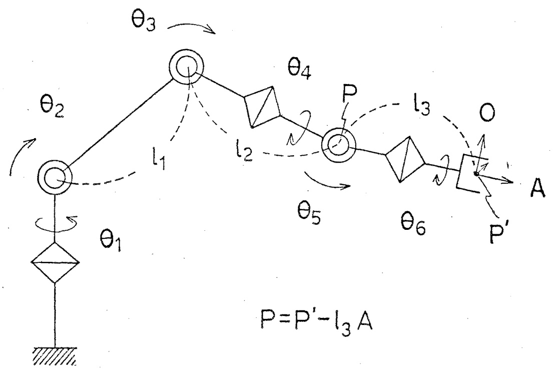


Fig. 1 A robot manipulator (6 degree of freedom)

られず，幾何学的な意味を考慮して，事例に即して解く必要がある．この解析解を求めるには，同次変換を用いて解く方法が有効であると言われている[6,7]．

この方法により，例えば **Fig. 1** のような6自由度多関節型ロボットの逆変換式を求めた結果は次の通りである．

$$\theta_1=\tan^{-1}\frac{P_y}{P_x} \tag{4}$$

$$\theta_3=\tan^{-1}\frac{\sqrt{(2l_1l_2)^2-(P_x{}^2+P_y{}^2+P_z{}^2-l_1{}^2-l_2{}^2)^2}}{P_x{}^2+P_y{}^2+P_z{}^2-l_1{}^2-l_2{}^2} \tag{5}$$

$$\theta_2=\tan^{-1}\frac{-P_z}{\sqrt{P_x{}^2+P_y{}^2}}+\tan^{-1}\frac{-l_2S_3}{l_1+l_2C_3} \tag{6}$$

$$\theta_4=\tan^{-1}\frac{-S_1A_x+C_1A_y}{-S_{23}(C_1A_x+S_1A_y)-C_{23}A_z} \tag{7}$$

$$\theta_5=\tan^{-1}\frac{\begin{matrix}-C_4(S_{23}(C_1A_x+S_1A_y)+C_{23}A_z)\\+S^4(-S_1A_x+C_1A_y)\end{matrix}}{C_{23}(C_1A_x+S_1A_y)-S_{23}A_z} \tag{8}$$

$$\theta_6=\tan^{-1}\frac{\begin{matrix}-C_5(C_4(-S_{23}(C_1O_x+S_1O_y)-C_{23}O_z)\\+S_4(-S_1O_x+C_1O_y))\\+S_5(C_{23}(C_1O_x+S_1O_y)-S_{23}O_z)\end{matrix}}{\begin{matrix}S_4(S_{23}(C_1O_x+S_1O_y)+C_{23}O_z)\\+C_4(-S_1O_x+C_1O_y)\end{matrix}} \tag{9}$$

但し，

$S_k=\sin\theta_k$

$C_k=\cos\theta_k$

$S_{23}=\sin(\theta_2+\theta_3)$

$C_{23}=\cos(\theta_2+\theta_3)$

$P=(P_x, P_y, P_z)$：手首の直交座標（Fig. 1 参照）

$A=(A_x, A_y, A_z)$：手先の接近（approach）を示す単位ベクトル

$O=(O_x, O_y, O_z)$：手先の方向（orientation）を示す単位ベクトル

このように，逆変換式は三角関数や逆三角関数等の特殊関数を含み，各関節で共通な演算も少ないことから，効率よく計算することが困難であった．従って，このような逆変換式を効率よく，高速に計算するアルゴリズムの開発が必要とされている．

2.2 逆変換演算の高速アルゴリズム

本論文では，逆変換の演算が主に座標の回転移動と逆正接により成り立っていることに着目し，これらの組み合わせで逆変換を行うことにより，演算の高速化を図った高速アルゴリズムを考案している．このように逆変換を，回転移動と逆正接の組み合わせで表すためには，与えられた解析解を1つ1つの回転移動に分解しなければならないが，以下のような手順に従えば容易に分解することができる．

i）逆変換式を分子分母に分け，それぞれを与式とする．また，$i=n$ とする．ただし，n は自由度，i は繰返しのインデックスを示し，G は θ の関数を表している．

ii）与式が，

$$\left.\begin{matrix}\cos\theta_iG_1(\theta)-\sin\theta_iG_2(\theta)\\ \sin\theta_iG_1(\theta)+\cos\theta_iG_2(\theta)\end{matrix}\right\} \tag{10}$$

の形に変形できるかを調べる．

(10) 式のように変形できる場合は，与式を座標 $(G_1(\theta), G_2(\theta))$ の θ_i に関する回転移動で表現する．

$$\begin{pmatrix}C_iG_1-S_iG_2\\S_iG_1+C_iG_2\end{pmatrix}=\begin{pmatrix}C_i & -S_i\\S_i & C_i\end{pmatrix}\begin{pmatrix}G_1\\G_2\end{pmatrix} \tag{11}$$

さらに，$G_1(\theta)$，$G_2(\theta)$ をそれぞれ新たな与式と定義し直し，手順iii）に進む．

(10) 式のように変形できない場合は，与式はそのまま手順iii）に進む．

iii）i を $i=i-1$ とし，i が0でなければ，手順ii）に戻る．i が0であれば，逆変換式の回転移動による分解を終了する．

ここでは，Fig. 1 の6自由度垂直多関節型マニピュレータの第4関節角 θ_4 を例にとり，上述の手順に従って具体的に分解してみる．

θ_4 は（7）式より

$$\theta_4=\tan^{-1}\frac{-S_1A_x+C_1A_y}{-S_{23}(C_1A_x+S_1A_y)-C_{23}A_z} \tag{12}$$

まず分母については，$i=2$, 3 つまり $(\theta_2+\theta_3)$ に関して，

$$\left.\begin{matrix}-(\sin(\theta_2+\theta_3)\cdot G_1+\cos(\theta_2+\theta_3)\cdot G_2)\\ G_1=A_x\cos\theta_1+A_y\sin\theta_1\\ G_2=A_z\end{matrix}\right\} \tag{13}$$

と表すことができるので，これを回転移動で表現する．

$$\begin{pmatrix}C_{23}G_1-S_{23}G_2\\S_{23}G_1+C_{23}G_2\end{pmatrix}=\begin{pmatrix}C_{23} & -S_{23}\\S_{23} & C_{23}\end{pmatrix}\begin{pmatrix}G_1\\G_2\end{pmatrix} \tag{14}$$

次に，新たにできた項 G_1 について $i=1$ として調べると，G_1 は

$$\left.\begin{array}{l} G_1=S_1G_3+C_1G_4 \\ G_3=A_y \\ G_4=A_x \end{array}\right\} \quad (15)$$

と表すことができ，これを回転移動により表現すると，

$$\begin{pmatrix} C_1G_3-S_1G_4 \\ S_1G_3+C_1G_4 \end{pmatrix}=\begin{pmatrix} C_1 & -S_1 \\ S_1 & C_1 \end{pmatrix}\begin{pmatrix} G_3 \\ G_4 \end{pmatrix} \quad (16)$$

となり分母の分解を終了する．

分子についても，同様に分解していけばよいが，この分子は（16）式で既に求められているので，これを用いることにする．

このようにして分解した回転移動を組み合わせることにより，（11）式の分子分母を次のように計算することができる．

$$\begin{pmatrix} A_Y \\ A_X \end{pmatrix}\xrightarrow{\theta_1 回転}\begin{pmatrix} -S_1A_X+C_1A_Y \\ C_1A_X+S_1A_Y \end{pmatrix}\to 分子$$

$$\begin{pmatrix} C_1A_X+S_1A_Y \\ A_Z \end{pmatrix}$$

$$\xrightarrow{\theta_2+\theta_3 回転}\begin{pmatrix} C_{23}(C_1A_X+S_1A_Y)-S_{23}A_Z \\ S_{23}(C_1A_X+S_1A_Y)+C_{23}A_Z \end{pmatrix}\to 分母$$

従って，（12）式は 16 回の演算が必要なところを 2 回の回転移動と 1 回の加算，1 回の逆正接のみで得られることがわかる．同様にして θ_5，θ_6 についても，上述の手順に従い，（8），（9）式を回転移動の組み合わせで表すことにより，演算量を減少させることができる．

このように逆変換の解析解を回転移動と逆正接で表すことにより，（4）～（9）式で必要とされる 83 回の演算を，25 回に減少させている．以上のようにして，逆変換の演算は回転移動を用いることにより，極めて効率よく計算することができる．本論文では，この回転移動の演算を高速に行う方法として，座標の回転を直接扱う CORDIC アルゴリズムを採用している．

2.3 CORDIC アルゴリズム

CORDIC アルゴリズムの概要を説明する．このアルゴリズムは，座標の回転移動を，離散的に繰り返し行い，収束させていくことで，任意の回転角に対する回転移動を得るものである．

一般に，2 次元直交座標（X_i, Y_i）の θ_j に関する回転移動は，

$$X_{i+1}=X_i\cos\theta_j-Y_i\sin\theta_j \quad (17)$$

$$Y_{i+1}=X_i\sin\theta_j+Y_i\cos\theta_j \quad (18)$$

で示される．ここで，i は繰返しのインデックスを表す．これらの式の両辺を $\cos\theta_j(\cos\theta_j\neq 0)$ で割ると

$$X_{i+1}/\cos\theta_j=X_i-Y_i\tan\theta_j \quad (19)$$

$$Y_{i+1}/\cos\theta_j=Y_i+X_i\tan\theta_j \quad (20)$$

となるが，ここで

$$\tan\theta_j=2^{-j} \quad (21)$$

となるように θ_i を選べば

$$X_{i+1}/\cos\theta_j=X_i-Y_i\cdot 2^{-j} \quad (22)$$

$$Y_{i+1}/\cos\theta_j=Y_i+X_i\cdot 2^{-j} \quad (23)$$

となり，θ_j に関する離散的な回転移動は，2 のべき乗の乗算，つまりシフトと加減算およびスケーリング補正（$\cos\theta_j$ 倍）によって演算できることになる．ここでは，(22)，(23) 式のスケーリング補正を後でまとめて行うことにし，（X_i, Y_i）から（X_{i+1}, Y_{i+1}）への移動を θ_j に関する回転移動およびスケーリング（$1/\cos\theta_j$ 倍）として新たに定義しなおすことにする．この移動は，シフトと加減算のみによって実行が可能である．この移動を回転方向も含めてまとめると，次のような基本演算で示される．

$$X_{i+1}=X_i-\delta_iY_i\cdot 2^{-j} \quad (24)$$

$$Y_{i+1}=Y_i+\delta_iX_i\cdot 2^{-j} \quad (25)$$

$$Z_{i+1}=Z_i-\delta_i\theta_j \quad (26)$$

ここで，δ_i は回転方向を示し，$\delta_i=1$ のとき正方向，$\delta_i=-1$ のとき負方向を表している．また，(26) 式の Z_i は回転角度の累積値を表わしている．

先に述べたように，この移動は θ_j だけ回転移動した後にスケーリング（$1/\cos\theta_j$ 倍）した座標を与えることになり，これを図で示すと **Fig. 2** のようになる．θ_j は (21) 式により **Table 1** のような値をとるが，これらを組み合わせることにより，任意の回転移動を実現している．この組み合わせの選び方，つまり j の取り方には特に制約はなく，自由に取ることができる．

座標の回転移動は，この基本演算において j を次式のようにとり

$$j=i-1 \qquad (i\geqq 1) \quad (27)$$

$Z_i\to 0$ となるように δ_i の符号を選び，$i=1$ から $n-1$ まで離散回転移動を行うことにより得られる（**Fig. 3**）．その結果は次式のように示される．

$$X_n=K(X_1\cos Z_1+Y_1\sin Z_1) \quad (28)$$

$$Y_n=K(-X_1\sin Z_1+Y_1\cos Z_1) \quad (29)$$

$$K=\prod_j(1/\cos\theta_j) \quad (30)$$

n は，基本演算の繰り返し回数を示しており，これを大きくとることにより，計算精度を上げることができる．K は n によってのみ決まる定数である．

このことをフローチャートで示すと，**Fig. 4** のようになる．このフローチャートでは，最初のルーチンで $\theta_0=90°$ の回転変換を付加することにより，$-180°$～$180°$ の範囲の回転変換を可能にしている．この $\theta_0=90°$ に対する基本式は，(17)．(18) 式に代入して

$$X_1=-\delta_0\cdot Y_0 \quad (31)$$

$$Y_1=\delta_0\cdot X_0 \quad (32)$$

になる．

また，基本式 (24)～(26) を用い，$Y_i\to 0$ となるよう

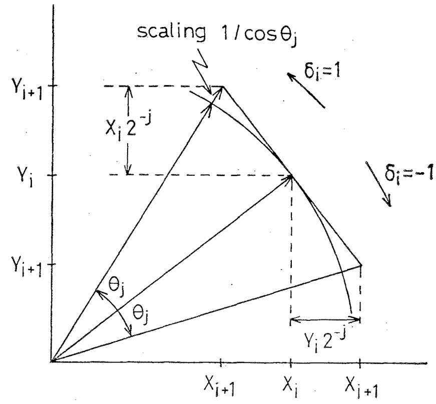


Fig. 2 The i-th step rotation in the CORDIC

Table 1 Discrete rotation angles

$\theta_0=45.000°$
$\theta_1=26.565°$
$\theta_2=14.036°$
$\theta_3=7.125°$
$\theta_4=3.576°$
$\theta_j=\tan^{-1}2^{-j}$

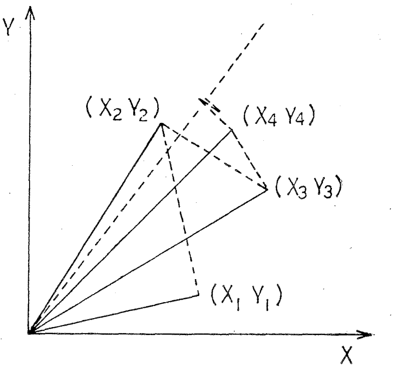


Fig. 3 Coordinate rotation in the CORDIC

に符号を選ぶことにより，逆正接およびベクトルの大きさを得ることができる．

$$X_n=K\sqrt{X_0^2+Y_0^2} \tag{33}$$

$$Z_n=-\tan^{-1}(Y_0/X_0)+Z_0 \tag{34}$$

このほか，次のような基本演算の繰り返しにより，乗算や平方根（$\sqrt{X_0^2-Y_0^2}$）を求めることができる．

乗算についての基本式は

$$X_{i+1}=X_i \tag{35}$$

$$Y_{i+1}=Y_i-\delta_i X_i\cdot 2^{-j} \tag{36}$$

$$Z_{i+1}=Z_i+\delta_i\cdot 2^{-j} \tag{37}$$

である．ここで j を次式を満たすようにとり

$$j=i+1 \qquad (i\geqq 0) \tag{38}$$

$Z_i\to 0$ となるように δ_i の符号を選んで，$i=0$ から $n-1$ まで繰返し演算することにより

$$Y_n=X_0\cdot Z_0+Y_0 \tag{39}$$

を得る．

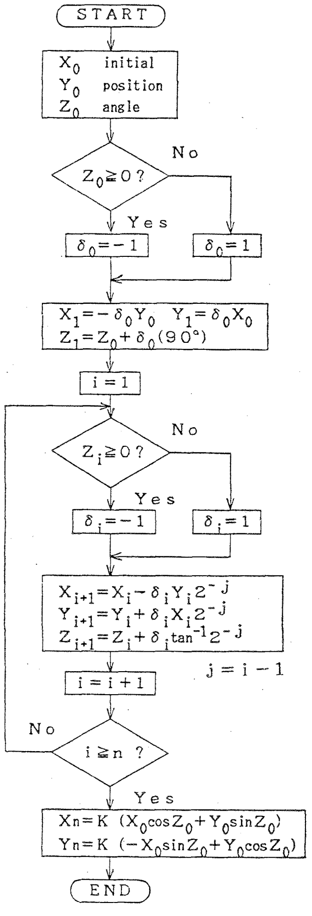


Fig. 4 Flowchart of the CORDIC operation (rotation mode)

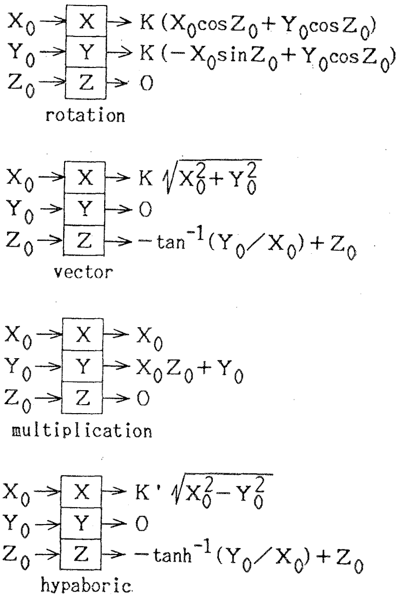
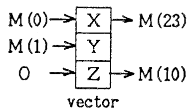


Fig. 5 CORDIC operation mode

(a) $\theta_1=\tan^{-1}\dfrac{Py}{Px}$

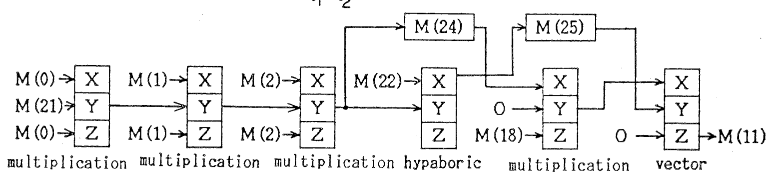


$M(0)=Px$　$M(1)=Py$　$M(2)=Pz$　$M(3)=-Pz$

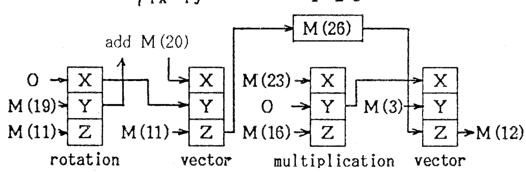
$M(4)=Ax$　$M(5)=Ay$　$M(6)=Az$　$M(7)=Ox$　$M(8)=Oy$　$M(9)=Oz$

$M(10)=-\theta_1$　$M(11)=-\theta_3$　$M(12)=-\theta_{23}$　$M(13)=-\theta_4$　$M(14)=-\theta_5$　$M(15)=-\theta_6$

(b) $\theta_3=\tan^{-1}\dfrac{\sqrt{(2l_1l_2)^2-(Px^2+Py^2+Pz^2-l_1^2-l_2^2)^2}}{Px^2+Py^2+Pz^2-l_1^2-l_2^2}$



(c) $\theta_2=\tan^{-1}\dfrac{-Pz}{\sqrt{Px^2+Py^2}}+\tan^{-1}\dfrac{-l_2S_3}{l_1+l_2C_3}$



$M(16)=1/K$

$M(17)=-1/K$

$M(18)=K'$

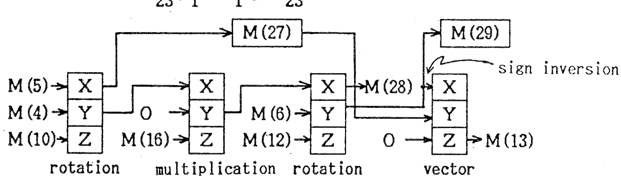
$M(19)=l_2$

$M(20)=K\cdot l_1$

$M(21)=-l_1^2-l_2^2$

$M(22)=2l_1l_2$

(d) $\theta_4=\tan^{-1}\dfrac{-S_1Ax+C_1Ay}{-S_{23}(C_1Ax+S_1Ay)-C_{23}Az}$



$M(23)=K\sqrt{Px^2+Py^2}$

$M(24)=Px^2+Py^2+Pz^2$

$M(25)=K'\sqrt{(2\ell_1\ell_2)^2-(Px^2+Py^2+Pz^2)^2}$

$M(26)=-\theta_3-\tan^{-1}(-S_3\ell_2/(\ell_1+C_3\ell_2))$

$M(27)=K(-S_1Ax+C_1Ay)$

$M(28)=K(C_{23}(C_1Ax+S_1Ay)-S_{23}Az)$

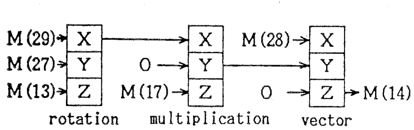
$M(29)=K(S_{23}(C_1Ax+S_1Ay)+C_{23}Az)$

$M(30)=K(-S_1Ox+C_1Oy)$

$M(31)=K(C_{23}(C_1Ox+S_1Oy)-S_{23}Oz)$

$M(32)=K^2(S_4(S_{23}(C_1Ox+S_1Oy)+C_{23}Oz)+C_4(-S_1Ox+C_1Oy))$

(e) $\theta_5=\tan^{-1}\dfrac{-C_4(S_{23}(C_1Ax+S_1Ay)+C_{23}Az)+S_4(-S_1Ax+C_1Ay)}{C_{23}(C_1Ax+S_1Ay)-S_{23}Az}$



(f) $\theta_6=\tan^{-1}\dfrac{-C_5(C_4(-S_{23}(C_1Ox+S_1Oy)-C_{23}Oz)+S_4(-S_1Ox+C_1Oy))+S_5(C_{23}(C_1Ox+S_1Oy)-S_{23}Oz)}{S_4(S_{23}(C_1Ox+S_1Oy)+C_{23}Oz)+C_4(-S_1Ox+C_1Oy)}$

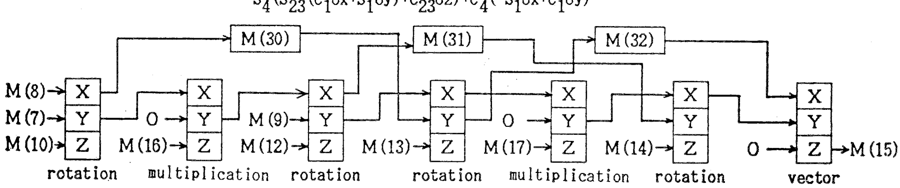


Fig. 6 Block diagram of the inverse transformation

双曲線関数についての基本式は

$$\tanh r_j = 2^{-j} \tag{40}$$

を満たすような r_j をとることにより

$$X_{i+1} = X_i + \delta_i Y_i \cdot 2^{-j} \tag{41}$$

$$Y_{i+1} = Y_i + \delta_i X_i \cdot 2^{-j} \tag{42}$$

$$Z_{i+1} = Z_i + \delta_i r_j \tag{43}$$

となる．j は i の関数となるが，ここで $j(i)$ を次式を満たすようにとり

$$r_j(i) \leqq \sum_{k=i-1}^{n-1} r_j(k) \tag{44}$$

$Y_i \to 0$ となるように δ_i の符号を選んで，$i=0$ から $n-1$ まで繰返し演算することにより

$$X_n = K'\sqrt{X_0^2 - Y_0^2} \tag{45}$$

$$Z_n = -\tanh^{-1}(Y_0/X_0) + Z_0 \tag{46}$$

$$K' = \prod_j (1/\cosh r_j) \tag{47}$$

を得る．

基本的な CORDIC 手法では，(40)～(47) 式で示される双曲線モードの演算において，X_0 と Y_0 の値が接近しているとき誤差が大きくなる性質があるので，これを防ぐために j の取り方を工夫する必要がある．(41)～(43) 式を計算するとき，j は (44) 式を満たすように取るのが基本的な方法であるが，このとき $\sqrt{X_0^2 - Y_0^2}$ が正確に求められる範囲は，例えば $X_0=1$ とすると，$|Y_0|<0.78$ である．この制限は，例えば（5）式のような双曲線モードを用いた逆変換演算において，処理できない作業空間領域を作ることになり，好ましくない．j は，(44) 式を満たすならばどのように選んでもかまわないので，ここでは，$j=1$ のループでの演算を繰り返し行い，Y_0 の範囲を広げることにより，逆変換が不可能な領域を減少させている．例えば，$j=1$ のループを4回繰り返すことにより，上の例で

$$|Y_0| < 0.99 \tag{48}$$

の範囲まで $\sqrt{X_0^2 - Y_0^2}$ を正確に求めることが可能である．

以上のような CORDIC 演算（回転移動，逆正接，乗算，双曲線関数）を **Fig. 5** のようなブロック図で示すことにする．

2.4 逆変換の具体例

ここで，各種のモードの CORDIC 演算を用い，Fig. 1 の6自由度垂直多関節型マニピュレータの逆変換処理の演算手順をブロック図で示すと，**Fig. 6** のようになる．

まず，θ_1 については（4）式より，

$$\theta_1 = \tan^{-1}\frac{P_y}{P_x} \tag{49}$$

これは，基本式において $X_0=P_x$，$Y_0=P_y$，$Z_0=0$ として，ベクトルモードで繰り返し演算を行い，結果として θ_1 を得る（Fig. 6（a））．

θ_3 については，

$$\theta_3 = \tan^{-1}\frac{\sqrt{(2l_1l_2)^2-(P_x^2+P_y^2+P_z^2-l_1^2-l_2^2)^2}}{P_x^2+P_y^2+P_z^2-l_1^2-l_2^2} \tag{50}$$

ここでは乗算モードを使い，$X_0=P_x$，$Y_0=-l_1^2-l_2^2$，$Z_0=P_x$ として，繰り返し演算を行い，

$$Y_n = P_x^2 - l_1^2 - l_1^2 \tag{51}$$

を得る．同様に積和演算を続けて，(50) 式の分母に相当する項 Q を得る．

$$Q = P_x^2 + P_y^2 + P_z^2 - l_1^2 - l_2^2 \tag{52}$$

分子については双曲線モードを使い，$X_0=2l_1l_2$，$Y_0=Q$，$Z_0=0$ として演算を行う．最後に，θ_1 と同様に逆正接を求め，θ_3 を得る（Fig. 6（b））．

θ_2 には三角関数の演算が含まれるので，これを求めるために回転モードを用いる（Fig. 6（c））．

θ_4～θ_6 については，その逆変換式が座標の回転移動の組み合わせから成り立っているため，回転移動を考慮しないで，直接的に演算を繰り返す方法に比べて極めて効率的に計算が可能である（Fig. 6（d），(e)，(f)）．

基本演算のときに示したように，回転モードでは結果に定数 K が乗じられるので，それぞれ $1/K$ をかけて補正する必要があるが，分子分母に K が含まれる場合は，これを既約化して乗算回数を2回減少させている．

3. LSI 向き座標逆変換プロセッサの構成

3.1 演算部

逆変換プロセッサの演算部は，2.3 節の CORDIC の基本式 (24)～(26)，(35)～(37)，(41)～(43) をハードウェアにより並列に演算することを可能としたものであり，**Fig. 7** のように加減算，シフタ，マルチプレクサ，レジスタなどから構成されている．また，RAM アクセスによる遅延の影響を避けるために，周辺にレジスタを配置し，基本演算部分を常時100％の稼働率で動作させている．構成要素を結ぶデータ信号はすべて並列に送られ，それぞれデータ語長だけの配線を必要とする．このデータの演算形式としては，固定小数点演算を用い，2の補数表現としている．

以上のような構成で Fig. 4 のフローチャートに従って基本演算を実行することにより，回転移動や逆正接などの演算を行う．2.3 節で説明したように，離散回転角 $\theta_0=90°$ のときの基本演算が他と異なっているが，本プロセッサでは，基本式 (24)，(25) の右辺第1項を0とするためにマルチプレクサを設けたり，シフタにおけるシフト量を0とすることにより，共通のルーチンで処理

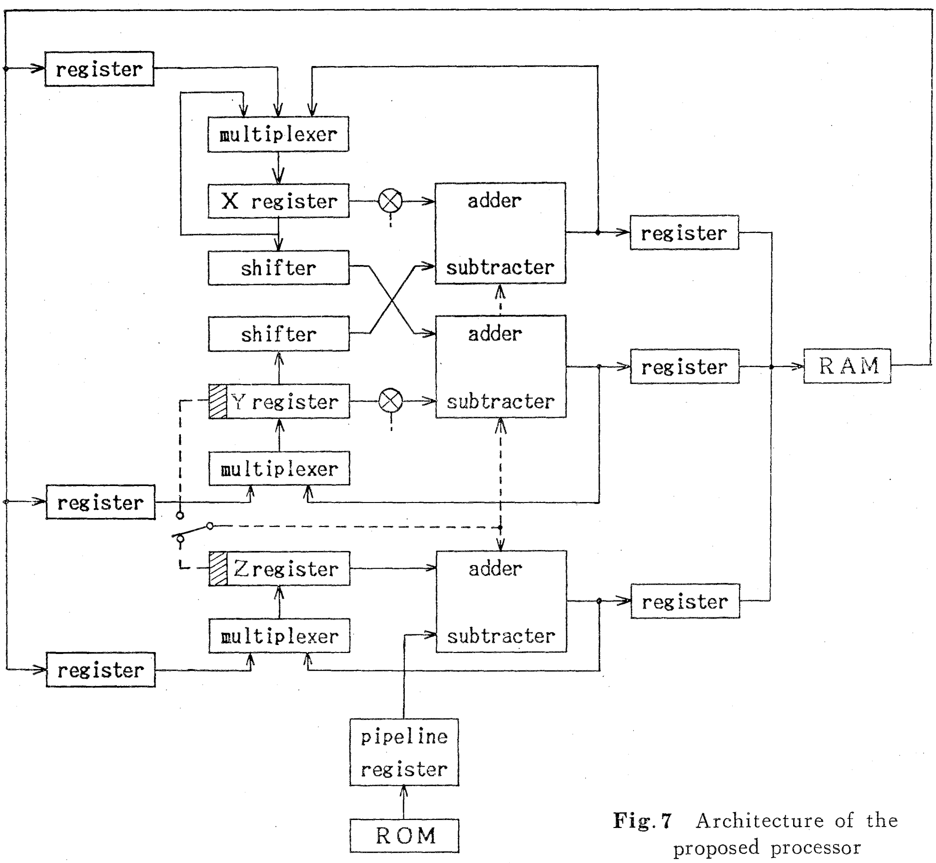


Fig. 7 Architecture of the proposed processor

を行っている.

$$X_{i+1}=X_i-\delta_i Y_i \cdot 2^{-j} \quad (53)$$

$$Y_{i+1}=Y_i+\delta_i X_i \cdot 2^{-j} \quad (54)$$

また，フローチャート中ベクトルモードでは Y_i の符号，回転モードでは Z_i の符号によって分岐が行われるが，データが2の補数表現であることから，Y レジスタあるいはZ レジスタの最上位ビットが0であるか，1であるかによって，シフタや定数ROMの出力を加算あるいは減算している.

乗算モードや双曲線モードの場合も，図中のマルチプレクサによりX レジスタの内容を保持したり，加減算のスイッチを制御することにより同一のハードウェアによる処理を可能にしている．このような構成により，新たに乗算器などを付加することなく，コンパクトな演算部を実現している.

3.2 制御部

制御部は，主にマイクロプログラム用ROMとカウンタ及びパイプラインレジスタにより構成され，このマイクロプログラムにより，演算部のモードスイッチや定数ROMのアドレス，シフト量などの制御を行っている．このマイクロプログラムを変更することにより，様々なマニピュレータに適用することができる.

Fig. 4 のフローチャートで説明したように，回転移動，逆正接などの演算を1回行うために，(24)～(26) 式のような基本演算のループを繰り返し実行しているが，制御部では，回転移動，逆正接などの演算を1つの単位として，制御を行うことにしている．このループでの制御は，初期座標や回転角度などの入力データ，回転変換後の座標や逆正接などの出力データを記憶するメモリアドレスの指定，演算モード選択の他は共通であり，これらをサブルーチン化することにより，プログラム量を減少し，プログラミングを容易にしている．これによれば，2.1 節のような逆変換演算は，Fig. 5 のブロック図をそのままコーディングすることにより，25 ステップのプログラミングのみで済むことになる.

4. 精度に関する評価

CORDIC 演算におけるデータの演算形式としては，固定小数点演算や浮動小数点演算が考えられる．ここでは，どちらの演算形式が有効であるかを検討するため，一例として，32 ビット固定小数点と 32 ビット浮動小数点（仮数部 24 ビット，指数部8ビット）を用いて，逆

正接演算をコンピュータでシミュレートし，両者の演算精度を比較している．CORDIC の基本演算は，(24)～(26) 式，(35)～(37) 式，(41)～(43) 式で示されるようにシフトと加減算によって成り，各モードで同様の演算が行われるため，それぞれの演算精度はほぼ等しいと考えられる．ここでは，CORDIC 演算の中で最も典型的な逆正接演算をとりあげている．

この逆正接演算のシミュレーションの結果を **Fig. 8** に示す．(a) では初期座標 (X_0Y_0) として，半径 0.5 の円周上の点を選び，偏角を変化させている．(b) では傾き 30° の直線上の点を選び，原点からの距離 $R(=\sqrt{X_0^2+Y_0^2})$ を変化させている．これらのグラフより，CORDIC 演算の精度は，回転角にはあまり関係なく，原点からの距離Rによってのみ決まることがわかる．このRが

$$0.01<R<1.0 \tag{55}$$

の範囲をとるとき固定小数点の方が精度が高いが，この理由としては，固定小数点の仮数部が，浮動小数点に比べて 8 ビット多いことや，CORDIC 演算が加減算の繰り返しから成っており，広いダイナミックレンジを必要としないことが考えられる．

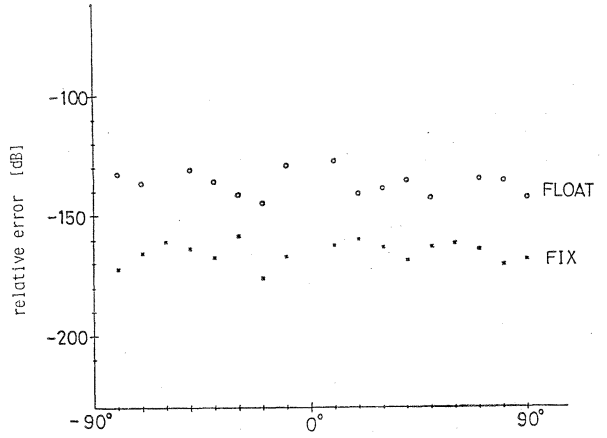
Rがそれ以下となる場合は，浮動小数点より精度が劣化してしまう．しかし，実際に座標逆変換を行う場合，マニピュレータの構造に起因する作業空間の制限により，Rはある範囲におさえられ，

$$R<0.01 \tag{56}$$

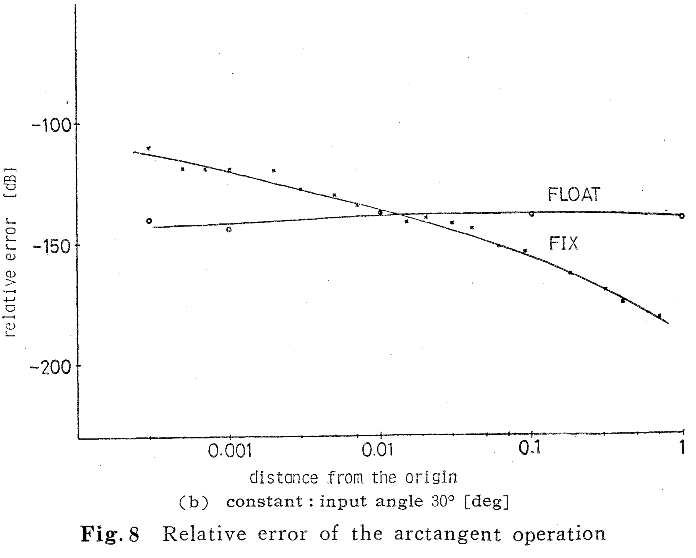
となることはまれであると考えられる．例えば，Fig. 1 のようなマニピュレータにおいて，腕の長さを 50 cm としたとき，台座付近の半径 1 cm の範囲を作業空間から除けば，Rは (55) 式の範囲に入り，固定小数点を用いた方が精度が高くなる．

以上のことより，実際の座標逆変換を考えた場合の CORDIC 演算は，固定小数点で行う方が高精度が得られることがわかる．

この他，ハードウェア量および処理速度の観点から，明らかに固定小数点演算は有利であるので，本プロセッサの演算形式としては，固定小数点演算を採用している．



(a) constant : distance from the origin 0.5



(b) constant : input angle 30° [deg]

Fig. 8 Relative error of the arctangent operation

5. ブレッドボードによる実験

動作確認および誤差解析を行うために，本プロセッサを Fig. 7 の構成に基づき，市販の個別 IC を用いてブレッドボード上に試作した．但し，データ長は 16 ビットで，基本演算部分を 20 ビットの固定小数点で，演算を行っている．基本演算を 20 ビットで行うのは，固定小数点加算を 16 回繰り返すことによる桁落ちを考慮してのことであり，こうすることにより 16 ビット精度を保証している．構成要素のシフタは，任意の右シフトを実行するものだが，便宜上シフトレジスタを用い必要なシフト演算を行っている．

試作機の制御は，語長 20 ビットのマイクロプログラムにより行われ，フィールドは **Table 2** に示す通りである．試

Table 2 Microprogram field

E	RAM address					register control									mode select		shift control		
N						internal				external									
D						X		Y	Z	in		out							
19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

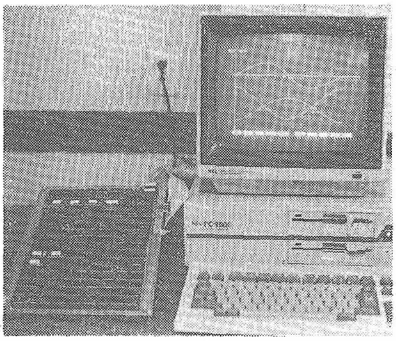


Fig. 9 Experimental system

作機では，3.2 節で述べたサブルーチン化を行っていないため，マイクロプログラムは約400ステップになっている．

実験では，入出力装置としてパーソナルコンピュータを用いており，試作プロセッサから出力される関節角度をディスプレイすることができる (**Fig. 9**)．これにより，Fig. 1 の6自由度多関節型ロボットの先端を，円弧や直線などからなる軌道に沿って動かしたときの逆変換を行い，試作プロセッサが正常に動作することを確認した．**Fig. 10** は，先端を (57)～(63) 式に従って動作させた場合の逆変換の結果を示している．

$$K=0\sim100 \tag{57}$$

$$\phi=K\pi/50 \tag{58}$$

$$P_X=0.3+0.1\cos\phi \tag{59}$$

$$P_Y=0.1\sin\phi \tag{60}$$

$$P_Z=0.004\,K \tag{61}$$

$$(A_x, A_y, A_z)=\left(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{-1}{\sqrt{3}}\right) \tag{62}$$

$$(O_x, O_y, O_z)=\left(\frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}\right) \tag{63}$$

このときの演算では，3番目の関節角 θ_3 が最も精度よく求まり，(64) 式で示される相対誤差は -86 dB であった．

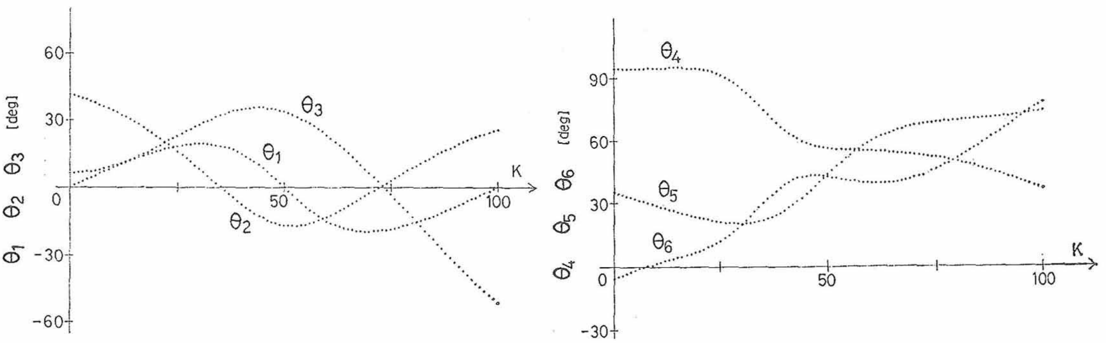
$$(\text{相対誤差})=20\log_{10}\frac{(\text{誤差})}{(\text{真値})} \tag{64}$$

これに対し，最も精度が悪かったのは θ_6 で，-56 dB であった．この理由は，固定小数点演算によるもので，θ_6 の値が他と比べて小さい値をとるために，相対誤差が劣化するものと思われる．

6. LSI 化に関する評価

本プロセッサの絶対遅れ時間について評価する．CORDIC による演算では，m ビットの精度を得るために，m回程度の基本演算（処理時間 τ）が必要であるから，1回の逆変換における絶対遅れ時間は $mN\tau$ となる．ここでNは，1回の逆変換に必要な演算の量でマニピュレータの形状によってきまる．

例えば，演算精度を $m=32$ ビットとしたとき，各構成要素の動作速度は，集積回路解析プログラム SPICE 2 により 2 μm CMOS LSI を前提としたシミュレーシ



(a) θ_1, θ_2, θ_3　　(b) θ_4, θ_5, θ_6 : $A=\left(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}\right)$, $O=\left(\frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}\right)$

Fig. 10 Result of the inverse transformation in the implemented processor

Table 3 Delay time obtained by the SPICE simulation (2 μm CMOS)

	Shifter T_{shift}	Adder T_{add}	Multiplexer T_{mux}
$X \cdot Y$	3 nS	17 nS	0.7 nS
Z		17 nS	0.7 nS

ョンを行った結果，**Table 3**注) のようになる．従って，基本演算の処理時間τは，定数 ROM のアクセスタイムを T_{rom} とすると，

$$\tau=\max(T_{\mathrm{shift}}, T_{\mathrm{rom}})+T_{\mathrm{add}}+T_{\mathrm{mux}} \qquad (65)$$

となるが，通常の場合 $T_{\mathrm{shift}} \ll T_{\mathrm{rom}}$ であるため，τは主に T_{rom} によって決定される．さらに，ROM の出力にパイプラインレジスタを設置し，加減算と ROM アクセスを並列に行うことにより，$\tau=T_{\mathrm{rom}}$ となる．ここで，$T_{\mathrm{rom}}=100$ nS とすれば，Fig. 1 の垂直多関節型ロボットの場合，演算量が $N=25$ であることから，絶対遅れ時間は $T=80\ \mu s$ となり，従来にはない極めて優れた性能が得られることがわかる．

次に，LSI 化を行う場合に必要となるトランジスタ数を評価する．本プロセッサの演算部は Fig. 6 の通りであり，加減算に BCLA (Block Carry Lookahead Adder)[9]，シフタにバレルシフタ[10]を用いる．また，レジスタにダイナミックレジスタ[10]を用いることにより，トランジスタ数を減少させている．これに周辺のマルチプレクサやレジスタを含めたトランジスタ数は，演算精度を 32 ビットで CMOS のとき，約1万4,000個と概算される．さらに，マイクロプログラム用と定数用の ROM 4 K ビット，ワークエリア用 RAM 1 K ビットを加えても，8 ビットシングルチップマイクロコンピュータと同程度の規模であり，現在の LSI 技術をもってすれば，本プロセッサのワンチップ化は，容易であると考えられる．

7. む　す　び

従来にない高速性やコンパクト性，様々なマニピュレータに適用可能な汎用性を備えた，座標逆変換プロセッサの構成や LSI 化の評価について述べた．本プロセッサ LSI が実現すれば，ロボット制御の様々な分野に利用可能であると考えられる．例えば，産業用ロボットの場合，作業座標系で教示が行われるため関節座標系に変換する必要があるが，従来はこの逆変換の処理時間が長いため，オフラインで処理されるのが一般的であった．本プロセッサを用いれば，逆変換をオンラインで処理することができ，教示の実時間処理や実時間の軌道修正が，容易に行えるようになる．また，マニピュレータの動作データを，関節座標系における時系列で蓄える必要がなく，大幅なデータメモリ量の削減が図れる．以上のような特長は，知能ロボットにおいてますます有用となると考えられる．現在，筆者らは LSI 化を前提として，汎用性，すなわちユーザが任意のロボットに対して使用容易なハードウェア構成やソフトウェアを検討中であり，超高性能座標逆変換プロセッサとして，十分実用性の高い LSI を開発できるとの見通しを得ている．

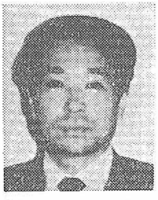
注) SPICE 2 パラメータに関しては，M社で実際に試作されている 2 μm CMOS プロセスの実測データを利用している．

参 考 文 献

1) 亀山充隆，樋口龍雄，“LSI 向きロボット制御用プロセッサ”，計測と制御，Vol. 25，No. 1，pp. 30–36，1986
2) 江上秀樹，亀山充隆，樋口龍雄，“ロボットマニピュレータ制御用超高速座標逆変換プロセッサの LSI アーキテクチャ”，第4回日本ロボット学会学術講演会予稿集，2208，pp. 243–244，12 月，1986
3) Andrew A. Goldenberg, B. Benhabib, Robert G. Fenton, “A Complete Generalized Solusion to the Inverse Kinematics of Robots”, IEEE Journal, Robots and Automation, Vol. RA-1, No. 1, pp. 14–20, 1985
4) 亀谷雅嗣，渡部　透，河田健一，鐵屋克浩，“マルチマイクロプロセッサによるロボット関節角計算の高速化”，日本ロボット学会誌，Vol. 3，No. 4，pp. 263–276，1985
5) Jack E. Volder, “The CORDIC Trigonometric Computing Technique”, IRE Trans., Eleectron. Comput., EC-8, pp. 330–334, 1959
6) Richard P. Paul，吉川恒夫訳，“ロボットマニピュレータ”，pp. 62–81，コロナ社，1984
7) Richard P. Paul, Bruce Shimano, Goldon E. Mayer, “Kinematic Control Equation for Simple Manipulators”, IEEE Trans., SMC-11, No. 6, pp. 449–455, 1981
8) J. C. Lai, “HSPICE user's manual Honeywell circuit simulation program”, Solid State Electronics Division, 1981
9) Kai Hwang, “Computer Arithmetic”, pp. 84–91, John Wiley & Sons, Inc., 1979
10) Carver Mead, Lynn Conway, “Introduction to VLSI Systems”, Addison Wesley Publishing Company, pp. 70–76, pp. 157–162, 1980

亀山充隆
(Michitaka KAMEYAMA)

昭和25年5月12日生．昭和53年，東北大学大学院工学研究科電子工学専攻博士課程修了(工学博士)．同年同大学助手．昭和56年同助教授．現在に至る．多値論理システム，高信頼化システム，LSIアーキテクチャなどの研究に従事．1984年及び1985年 IEEE ISMVL 優秀論文賞受賞．昭和61年度計測自動制御学会技術賞受賞．IEEE などの会員．



樋口龍雄 (Tatsuo HIGUCHI)

昭和15年3月30日生．昭和37年東北大学工学部電子工学科卒業．昭和42年同大学院博士課程修了．工学博士．東北大学工学部助手，助教授を経て昭和55年同教授，現在に至る．この間，ディジタル信号処理，特にディジタルフィルタの統一的構成理論および信号処理プロセッサ，多値論理システムとその応用などの研究に従事．1984年，1985年，IEEE ISMVL 優秀論文賞，昭和59年度計測自動制御学会論文賞，昭和61年度同学会技術賞受賞．計測自動制御学会，電子通信学会，電気学会，IEEE などの会員．

江上秀樹 (Hideki EGAMI)

昭和38年4月22日生．昭和61年東北大学工学部電子工学科卒業．現在，同大学院修士課程在学中．ロボットエレクトロニクスに関する研究に従事．昭和62年度日本ロボット学会研究奨励賞受賞．電子情報通信学会准員．

(日本ロボット学会学生会員)

Design of an Ultra-High-Speed Inverse-Kinematic Processor for Robot Control*

Michitaka KAMEYAMA**
Hideki EGAMI** Tatsuo HIGUCHI**

ABSTRACT

This paper presents an LSI-oriented high-performance processor for the inverse kinematics to control the position and orientation of the end-effector of a robot. It is well known that a geometric approach is useful in finding such joint angles. To achieve the high-speed transformation, a new hardware algorithm based on the coordinate rotation is proposed which can be implemented by the CORDIC technique. By means of the algorithm, an inverse-kinematic processor is designed which has attractive features of high-speed, compactness, and flexibility for any kinds of manipulators. The processor has been implemented on the breadboard using TTL-IC's in order to confirm the operation. Assuming the 2 μm CMOS technology, the chip evaluation of the processor is discussed from the viewpoint of the speed and chip area. Although the hardware size is almost same as that of an 8-bit one-chip microcomputer, ultra-high-speed transformation can be achieved. As a result, it is established that the inverse kinematic processor is very effective for the practical applications.

Key words : Robot manipulator, Inverse kinematics, Special-purpose processor, CORDIC, LSI

* Received March 12, 1987
** Faculty of Engineering, Tohoku University