특집

Global Positioning System 응용을 위한 파이프라인 형 CORDIC 회로 설계

李 殷 均, 劉 泳 甲
忠北大學校 情報通信工學科

요 약

이 논문에서는 GPS의 고속 측지 알고리즘에 활용될 고속 삼각함수 계산회로를 제안하였으며, 그 시제작을 위하여 FPGA 를 활용한 예를 제시하였다. 기존의 삼각함수 계산에 사용되는 CORDIC 알고리즘을 파이프라인 구조로 구현하여 다량의 계산을 전체적으로 신속하게 수행할 수 있는 구조를 설계하였다. 이 파이프라인 구조는 계산결과의 정밀도의 요구에 따라 칩의 회로의 규모를 가변시킬 수 있도록 단계 슬라이스형 구조를 도입하였다. 제어회로와 연산회로를 모두 파이프라인 구조를 모두 단계 슬라이스 형으로 설계하였으며, 파이프라인 슬라이스의 개수에 따라 정밀도가 달라지게 하였다. 또한 FPGA 칩을 여러개 사용하여 전체 파이프라인이 구성되는 관계로 칩간 통신에는 더미 사이클을 도입하여 칩의 입출력에 필요한 시간을 확보하는 기법을 구사하였다.

Abstract

A new stage-sliced pipeline structure is presented to design a high speed pipelined CORDIC processor for high speed real time Global Positional Systems(GPS) applications. The CORDIC algorothm was revised to generate a pipeline structure, which will be used to produce a large amount of trigonometric computations rapidly. A stage-sliced approach was introduced to adjust the number of iterative processes, and thereby to control the precision of computation results. Both the computation and the control circuits of the proposed architecture are included in a pipeline stage, which are integrated into a stage slice. The circuit was prototyped using six FPGA chips : one is used for glue logics and five of the chips are used for

pipeline slice implementation. A single FPGA chip comprising 7 pipeline stages provides one pipeline slice. To compensate an inter-slice time delay, dummy cycles are introduced in inter-slice signal exchanges.

I. 서 론

자동 항법 장치 및 위치 판독 장치의 근간으로서 인공위성에서 발사되는 신호의 활용이 보편화되고 있다. 이 GPS(Global Positioning System)의 기반은 여러 위성에서 발사되는 신호를 식별하여 수신자의 위치를 정확하게 판독하여야 하는 알고리즘의 정확성에 있다. 이 위치 판독 알고리즘을 실시간 GPS에 응용하는 과정에서 주로 문제가 되는 것은 신속하고 정확한 삼각함수 값의 획득에 있다. 삼각함수 계산의 신속성 여하에 따라 이동형 GPS의 실시간 적용 범위가 결정되기 때문이다.

삼각함수 계산으로서 보편적으로 활용되어온 알고리즘으로서 CORDIC(Coordinate Rotation Digital Computer) 기술이 있다. 이는 소프트웨어에 의한 반복처리를 거쳐서 루트 계산, 사인, 코사인, 하이퍼블릭 사인, 하이퍼블릭 코사인, 역 탄젠트, 역 하이퍼블릭 탄젠트등의 특수함수를 계산하고 있다. 따라서 소프트웨어 알고리즘이 일반적으로 갖는 계산속도의 제약으로 인하여 고속의 위치 판독을 목표로하는 GPS 응용에는 어려움과 제약을 피할 수 없다.

고속 삼각함수 계산을 위하여 이 CORDIC 알고리즘을 하드웨어로 구현하는 과정에서 반복형 계산과정을 배열회로로써 구현하고 그 효과를 측정해보는 것이 GPS의 실시간 응용을 위하여 필요하다. 즉 반복계산에 필요한 최소한의 하드웨어를 설계하고 이를 필요한 횟수만큼 반복설계하여 그 각각을 상호연결하므로써 소프트웨어의 반복계산 효과를 하드웨어만으로 구현하는 것이다. 따라서 삼각함수 계산은 이들 하드웨어를 통과하는 과정에서 실행되며, 소프트웨어의 개입이 전혀 없으므로 획기적인 계산속도의 향상을 볼 수 있는 것이다.

또한 많은 양을 삼각함수 계산을 하드웨어로써 단기간내에 처리하기 위하여 이 배열형 구조를 파이프라인화하는 것이 효과적이다. 특히 계산알고리즘의 종료시 이외에는 분기명령이 없으므로 파이프라인 구조가 갖는 속도저하의 요인은 없다. 이 계산에 필요한 평균시간은 반복계산중 단 한단계의 회로를 통과하는데 걸리는 시간이 되며, 전체 삼각함수 계산을 수행할 수 있게 되는 것이다. 따라서 GPS의 위치 판독 계산에 필요한 많은 양의 삼각함수 계산을 이 파이프라인 구조의 배열형 회로로써 신속하게 처리할 수 있으므로, GPS의 실시간응용범위의 확대에 기여할 수 있는 것이다.

본 논문에서 CORDIC 알고리즘을 배열형 하드웨어로 구현하는 방법을 제시하였다. 구현된 파이프라인 구조를 갖는 배열형 CORDIC 회로는 제어부와 연산부를 갖는다. 제어부에서는 제어신호를 생성하며 연산부를 제어하고, 연산부에서는 sin 과 cos 값을 계산한다. 제어부의 구성은 30단의 배열로 구성되었으며, 마지막 두 단에서 cos과 sin 값을 각각 출력하도록 하였다. 이들은 FPGA 칩으로 나뉘어 설계되었다. 각 FPGA 칩에 분산된 파이프라인 단계는 계산결과의 정밀도의 요구에 따라 그 수를 조절할 수 있도록 하였다.

본 논문은 다음과 같이 구성된다. 2장에서는 GPS의 3차원 모델에서의 삼각함수 계산과정을 소개하고, 3장에서는 CORDIC 알고리즘을 이용한 파이프라인 구조를 갖는 배열형 CORDIC 설계와 칩 구현에 대해 소개한다. 4장에서는 측정 결과를 보이고, 끝으로 5장에서는 결론을 기술한다.

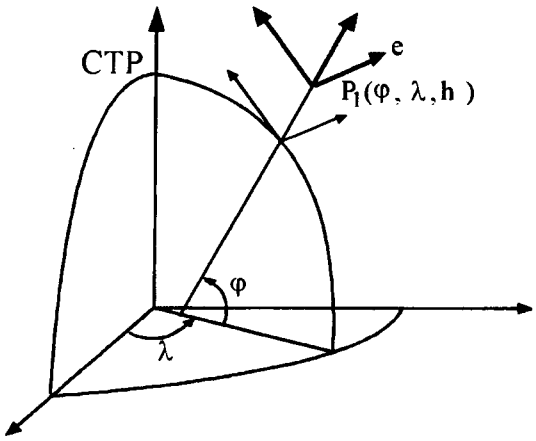
II. GPS를 위한 3-차원 측지 모델

먼저 GPS에서의 위치 판독을 위한 이론적 모델이 되는 3-차원 측지 모델을 살펴 보도록 하자. 이는 자동 항법장치 및 위치 판독장치의 근간이다. 본 논문은 3-차원 측지 모델에서 소요되는 삼각함

수 계산은 CORDIC 알고리즘의 하드웨어화로써 구현하게된다.

1. 국부 측지 좌표계(Local Geodetic Coordinate System)

그림 1은 수학적 모델의 발전에 중심적 역할을 했던 국부측지좌표계를 보여주고 있다. n축과 e축은 Local Geodetic Horizen 에 수평으로 놓여 있다. n축은 북쪽을 향하고 있으며, e축은 동쪽을 향하고 있다. h축은 타원과 수직으로 존재한다. 이때 국부측지좌표는 (n, e, h)이다. 국부측지좌표계에서 공간의 방위는 측지의 위도 φ와 측지의 경도 λ 에 의해 결정된다. w축은 Conventional Terrestrial Pole(CTP) 의 방향과 일치한다.



〈그림 1〉 국부 측지 좌표 시스템

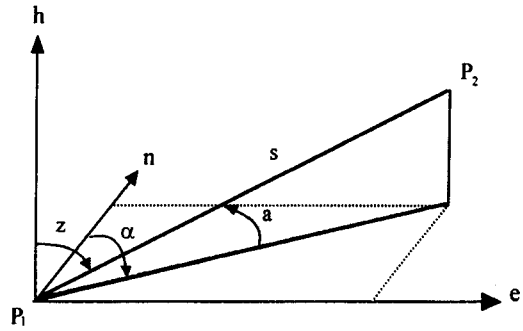
그림 2는 점 P_1 과 P_2 사이의 타원 평면과 수직각이 국부측지좌표계와 어떤 관계가 있는지를 보여주고 있다. 기호 a 는 타원의 고도각을 위해 사용하며, 기호 h 는 P_1 의 국부측지좌표계 에서 두 번째 점 P_2 까지의 높이를 의미한다. 각각을 수식으로 나타내면 다음과 같다.

$$n = s\cos a\cos a \tag{1}$$

$$e = s\cos a\sin a \tag{2}$$

$$h = s\sin a \tag{3}$$

식 1, 2, 3을 다시 정리하면 다음과 같다.



〈그림 2〉 국부측지좌표계의 결과 모델

$$a = \tan^{-1}\left(\frac{e}{n}\right) \tag{4}$$

$$a = \sin^{-1}\left(\frac{h}{s}\right) \tag{5}$$

$$s = \sqrt{n^2+e^2+h^2} \tag{6}$$

2. 측지 직교 좌표계(Geocentric Cartesian System)

국부측지좌표계와 Geocentric Cartesian System 과의 관계{U=(u, v, w)}는 그림 1에서 볼수 있다.

$$\begin{bmatrix} n \\ -e \\ h \end{bmatrix} = R_2(\varphi-90^0)R_3(\lambda-180^0)\begin{bmatrix} \Delta u \\ \Delta v \\ \Delta w \end{bmatrix} \tag{7}$$

이때, R_2와 R_3 는 다음과 같이 정의한다.

$$R_1(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix} \tag{8}$$

$$R_2(\theta) = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \tag{9}$$

$$R_3(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{10}$$

$$\begin{bmatrix} \Delta u \\ \Delta v \\ \Delta w \end{bmatrix} = \begin{bmatrix} u_2-u_1 \\ v_2-v_1 \\ w_2-w_1 \end{bmatrix} \tag{11}$$

식 7에서 e 의 부호를 변화시키고, 회전 행렬을 합하면, R_2 와 R_3 는 다음과 같다.

$$\begin{bmatrix} n \\ e \\ h \end{bmatrix} = R(\varphi,\ \lambda) \begin{bmatrix} \Delta u \\ \Delta v \\ \Delta w \end{bmatrix} \tag{12}$$

이때, R을 전개하면, 다음과 같다.

$$R = \begin{bmatrix} -\sin\varphi\cos\lambda & -\sin\varphi\sin\lambda & \cos\varphi \\ -\sin\lambda & \cos\lambda & 0 \\ \cos\varphi\cos\lambda & \cos\varphi\sin\lambda & \sin\varphi \end{bmatrix} \tag{13}$$

식 4, 5, 6, 12, 13으로 부터 다음과 같은 수식을 얻을수 있다. 이 수식은 Geocentric Cartesian Coordinate 차이와 P_1 의 Geodetic Position의 함수를 계산하기 위한 수식으로 이용된다.

$$a_1 = \tan^{-1}\left(\frac{-\sin\lambda_1\Delta u + \cos\lambda_1\Delta v}{-\sin\varphi_1\cos\lambda_1\Delta u - \sin\varphi_1\sin\lambda_1\Delta v + \cos\varphi_1\Delta w}\right) \tag{14}$$

$$a_1 = \sin^{-1}\left(\frac{\cos\varphi_1\cos\lambda_1\Delta u + \cos\varphi_1\sin\lambda_1\Delta v + \sin\varphi_1\Delta w}{\sqrt{\Delta u^2 + \Delta v^2 + \Delta w^2}}\right) \tag{15}$$

$$s = \sqrt{\Delta u^2 + \Delta v^2 + \Delta w^2} \tag{16}$$

이 수식에서 Geodetic 위도와 경도는 P_1 점과 관계가 있다. 식 14, 15, 16은 $La = F(Xa)$ 으로 표시되는 observation equation type 의 모델로 나타낼 수 있다.

3. 3-차원 측지모델

3차원 측지모델의 계산과정에서 least-squares adjustment 에 의해 추정되는 변수는 (Δu, Δv, Δw)이다. 이에 관한 비선형 모델은 다음과 같은 일반적인 형태를 갖는다.

$$a_1 = f(u_1,\ v_1,\ w_1.\ u_2,\ v_2,\ w_2) \tag{17}$$

$$a_1 = f(u_1,\ v_1,\ w_1.\ u_2,\ v_2,\ w_2) \tag{18}$$

$$s = f(u_1,\ v_1,\ w_1.\ u_2,\ v_2,\ w_2) \tag{19}$$

원하는 행렬에 적당한 값을 찾기 위해, 파라미터가 의미하는 전체 편 도함수를 요구한다.

일반적인 형태는 다음과 같다.

$$\begin{bmatrix} da_1 \\ da_1 \\ ds \end{bmatrix} = \begin{bmatrix} g_{11} & g_{12} & g_{13} : g_{14} & g_{15} & g_{16} \\ g_{21} & g_{22} & g_{23} : g_{24} & g_{25} & g_{26} \\ g_{31} & g_{32} & g_{33} : g_{34} & g_{35} & g_{36} \end{bmatrix} \begin{bmatrix} du_1 \\ dv_1 \\ dw_1 \\ du_2 \\ dv_2 \\ dw_2 \end{bmatrix}$$

$$= [G_1 : G_2] \begin{bmatrix} dU_1 \\ dU_2 \end{bmatrix} \tag{30}$$

이 때, g에 대한 편도함수는 아래 표 1 와 같다.

GPS를 항공기, 차량등 이동형장비에 탑재하고 실시간으로 위치판독을 수행하기 위하여, 고속 곱셈 및 나눗셈 계산과 함께 고속 삼각함수계산이

〈표 1〉 직교좌표의 항으로 전개된 3-차원 측지 모델의 편도함수

$$g_{11} = \frac{\partial a_1}{\partial u_1} = -g_{14} = \frac{-\sin\varphi_1\cos\lambda_1\sin a_1 + \sin\lambda_1\cos a_1}{s\ \cos a_1} \tag{a}$$

$$g_{12} = \frac{\partial a_1}{\partial v_1} = -g_{15} = \frac{-\sin\varphi_1\cos\lambda_1\sin a_1 + \sin\lambda_1\cos a_1}{s\ \cos a_1} \tag{b}$$

$$g_{13} = \frac{\partial a_1}{\partial w_1} = -g_{16} = \frac{\cos\varphi_1\sin\lambda_1}{s\ \cos a_1} \tag{c}$$

$$g_{21} = \frac{\partial a_1}{\partial u_1} = -g_{24} = \frac{-s\cos\varphi_1\cos\lambda_1 + \sin a_1\Delta u}{s^2\ \cos a_1} \tag{d}$$

$$g_{22} = \frac{\partial a_1}{\partial v_1} = -g_{25} = \frac{-s\cos\varphi_1\sin\lambda_1 + \sin a_1\Delta v}{s^2\ \cos a_1} \tag{e}$$

$$g_{23} = \frac{\partial a_1}{\partial v_1} = -g_{26} = \frac{-s\sin s\varphi_1 + \sin a_1\Delta w}{s^2\ \cos a_1} \tag{f}$$

$$g_{31} = \frac{\partial s_1}{\partial u_1} = -g_{34} = -\frac{\Delta u}{s} \tag{g}$$

$$g_{32} = \frac{\partial s_1}{\partial v_1} = -g_3 = -\frac{\Delta v}{s} \tag{h}$$

$$g_{33} = \frac{\partial s_1}{\partial v_1} = -g_{36} = -\frac{\Delta w}{s} \tag{i}$$

요구된다. 하나의 위치좌표 계산에는 최소한 6 회의 삼각함수 값이 요구되며, 이 값이 주어지고 난 후에야 다른 계산을 시작할 수 있다. 이러한 고속 계산의 요구는 다음에 설명될 배열형 CORDIC 파이프라인으로써 충족될 수 있다.

III. 파이프라인 구조의 배열형 CORDIC설계

앞에서 설명한 GPS의 실시간 처리환경에서 요구되는 핵심 계산은 삼각함수의 고속처리이다. 이를 위하여 먼저 기존의 배열형 CORDIC의 원리와 설계에 대해서 살펴본다. 먼저 기존의 CORDIC 알고리즘의 원리[3]와 구조에 대해서 알아보고, 이 논문에서 새로 제안한 배열형 CORDIC 원리와 구조를 제시한다. 그리고 계산의 고속화를 위하여 제안된 배열형 CORDIC의 설계 및 구현을 제시하고자 한다.

1. CORDIC의 원리

먼저 임의의 벡터 (R,β)의 회전을 고려해보자. 기본 벡터를 X와 Y를 요소로 하는 직각 좌표로 가정하자. 이 기본벡터를 양각 θ 만큼 회전시킨 벡터 X'와 Y'을 구하려고 한다.

그림 3(a) 에서 보면 다음과 같은 관계를 얻게 된다.

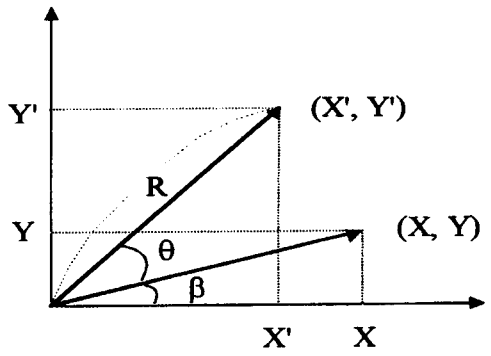
$$X' = X\cos\theta - Y\sin\theta \quad (31)$$

$$Y' = X\cos\theta - Y\sin\theta \quad (32)$$

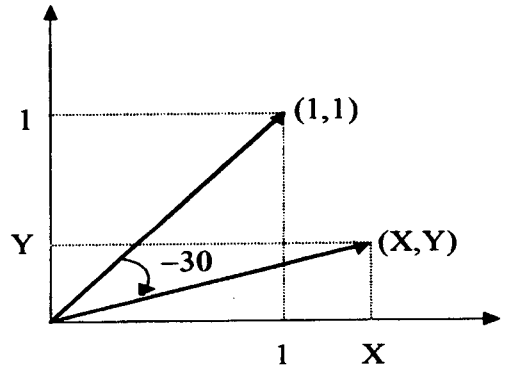
$$\frac{X'}{\cos\theta} = X - Y\tan\theta \quad (33)$$

$$\frac{Y'}{\cos\theta} = Y + X\tan\theta \quad (34)$$

CORDIC 알고리즘은 각 단계의 벡터를 회전각 크기 $a_i = tan^{-1}2^{-i}$만큼 시계방향 혹은 반시계 방향으로회전시킨다.각a_i의 +혹은 -방향은$(\theta - \Sigma a_j)$에 따라 선택되어진다. 이때 $(\theta - \Sigma a_j) \rangle 0$이면 +방향이고 $(\theta - \Sigma a_j) \langle 0$ 이면 -방향이 된다. θi 는 단계 i 에서 회전된 벡터가 가지고 있는 각의 크기를



(a) 회전각 θ



(b) 회전각 $-30°$

〈그림 3〉 회전각 θ를 갖는 벡터 (R,β)

총합한 것이다. 즉, $\theta_i = \pm\alpha_0 \pm \alpha_1 \pm \alpha_2 \cdots \pm \alpha_i$이다. θ_i 의 최대 크기가 100° 이하이므로 $|\theta| > 90°$인 각은 우선 90° 이하 크기로 조정된다면 특정 각에 대해 접근은 $\pm\alpha_i$ 각만큼의 반복적인 회전 과정을 통해 이루어질 수 있다.

$$X_{i+1} = X_i - Y_i\tan\alpha_i = X_i \mp Y_i \times 2^{-i} \quad (35)$$

$$Y_{i+1} = Y_i - X_i\tan\alpha_i = Y_i \mp X_i \times 2^{-i} \quad (36)$$

여기에서 $i = 0, 1, 2, \cdots$ 이고 X_0와 Y_0는 기본 벡터의 X와 Y값이다. 식 33과 34로 부터 기인한 35와 36에서의 X_{i+1} 과 Y_{i+1} 은 (X_i, Y_i) 벡터보다 $1/\cos\alpha_i$비율로 큰 값이 된다. 단계별 $1/\cos\alpha_i$ 을 보정하기 위해 계산하면 임의의 상수 K_c을 정의할 수 있다.

〈표 2〉 백터(X=1, Y=1)의 −30 ° 회전시 변화 단계표

i	$\tan \alpha_i$	α_i	θ_i	$\theta-\theta_i$	Y_{i+1}	Y_{i+1}
				-30°	1	1
0	1	-45°	-45°	+15°	2	0
1	2^{-1}	+26.6°	-18.4°	-11.6°	2	1
2	2^{-2}	-14.04°	-32.44°	+2.44°	2.25	0.5
3	2^{-3}	+7.13°	-25.31°	-4.69°	2.187	0.781
4	2^{-4}	-3.58°	-28.89°	-1.11°	2.236	0.644
5	2^{-5}	-1.79°	-30.68°	+0.68°	2.256	0.574
6	2^{-6}	+0.90°	-29.78°	-0.22°	2.247	0.609
7	2^{-7}	-0.45°	-30.23°	+0.23°	2.252	0.591
8	2^{-8}	+0.22°	-30.01°	+0.01°	2.250	0.600
9	2^{-9}	+0.11°	-29.90°	-0.10°	2.248	0.604

$$K_c=\frac{1}{\cos\alpha_0}\cdot\frac{1}{\cos\alpha_1}\cdot\frac{1}{\cos\alpha_2}\cdot\cdots\cdots$$
$$\sqrt{(1+2^{-1})}\times\sqrt{(1+2^{-2})}\times\sqrt{(1+2^{-4})}\times\cdots\cdots$$
$$1.646760255\cdots\cdots \quad (37)$$

식 37에서 얻어진 값은 i = 24일 때, 즉 24회 반복계산한 값이다.

위 CORDIC알고리즘을 이용하여 $\cos\theta$와 $\sin\theta$를 구해 보도록 하겠다. 그림 3(b)는 벡터($\sqrt{2}$, 45°)가 -30°만큼 회전한 것이며 우리는 변화한 X와 Y값을 구할 수 있게 된다. 식 35와 36을 이용하여 마지막 단계의 X와 Y값을 구하므로써 $\cos\theta$와 $\sin\theta$를 계산 할 수 있다.

표 2은 기본 벡터 (X=1, Y=1)가 각 -30° 회전하는 동안에 변화하는 X_i, Y_i를 보여 준다. 90° 가 넘는 회전각은 우선 각 조정을 한 후 회전시킨다. 그러나 이 단계는 표 2에서는 포함하지 않았다. 이러한 반복 과정을 통하여 얻어지는 마지막 값 X와 Y를 식 38, 39와 같이 예상할 수 있다.

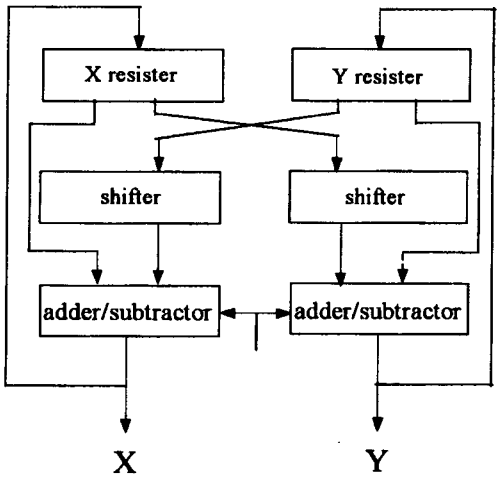
$$X=\sqrt{2}\cos 15^{\circ}\times K_c\simeq 2.24951634\cdots \quad (38)$$

$$Y=\sqrt{2}\sin 15^{\circ}\times K_c\simeq 0.60275609\cdots \quad (39)$$

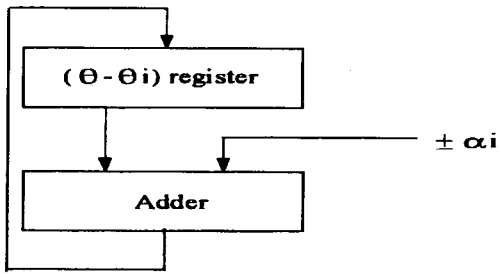
X와 Y의 값은 물론 상수 K_c로 나눠야하지만(초기값 X와 Y값을 K_c로 나눌 수도 있다.) 하드웨어 설계시 미리 K_c로 나눈 기본 벡터값을 이용한다면 구조가 더욱 간단하면서 구현이 용이하다.

CORDIC 알고리즘의 구현은 식 35와 36을 이용하게 되며 본 논문에서도 역시 이 식을 이용하여 기본 벡터을 변형하게 된다. 식 35와 36을 이용하여 각 단계의 Xi 와 Yi 의 값을 구하기 위해서는 크게 값을 직접 계산하는 연산부와 연산부를 제어하는 제어부로 나뉘어지며 아래 그림 4와 같다.

연산부는 그림 4(a) 이며 X_i 와 Y_i 의 레지스터



(a) logic 블럭



(a) logic 블럭

〈그림 4〉 반복형 CORDIC의 구조

와 각 단계의 쉬프터와 두개의 가산기/감산기가 필요하다. 제어부는 가산기와 감산기중에서 한 동작을 선택하는 기능을 한다. 제어부는$(\theta-\theta_i)$ accumulater와 $\pm\alpha_i$ 를 저장하는 ROM, 가산기로 이루어진다. α_i의 부호는 $(\theta-\theta_i)$ 레지스터의 부호 비트에 의해 결정되는데 α_i 의 값은 일반적으로 ROM에 저장된다. 그림 4(b) 는 이러한 요소의 블럭도를 보여준다. 그림 4(a) 에서 교차된 경로는 식 35와 36에 의해 가산 혹은 감산될 X_i와 Y_i의 쉬프트 된 값을 운반한다. 여기서 사용되는 쉬프터는 barrel shifter로서 구현에는 회로 설계의 부담이 있다. 그러나 본 논문에서와 같이 배열화 할 때에 논리 단계가 간단하여 쉽게 해결할 수 있으므로, 각종 프로세서의 부속 회로화가 용이하다.

2. 제안된 파이프라인 구조의 배열형 CORDIC 설계

실시간 GPS의 응용을 위하여는 계산의 정밀도와 계산시간의 단축이 중요한 문제이다. 계산의 정밀도는 소수점 이하 9자리 정도를 필요로하며, 이를 위하여는 32비트의 데이터 표현이 요구된다. 계산량도 여러개의 함수값을 동시에 계산하여야하므로 한 개의 데이터에대한 삼각함수 계산보다는 한 묶음의 데이터에 대한 총 계산시간이 문제가 된다. 여기에는 파이프라인 구조가 가장 적합한 해결책이 된다. 따라서 32비트 연산을 기본으로하는 배열형 파이프라인 CORDIC을 설계하게 된다.

본 논문에서 제안된 배열형 CORDIC 회로는 CORDIC 알고리즘의 K_c 값을 이용하여 출력값이 $\cos\theta$와 $\sin\theta$가 되도록 하였으며, 같은 회로의 반복 사용을 하는 기존의 방법에서 루프선을 없애고 배열화하여 속도를 향상시켰다. 우선 CORDIC 알고리즘의 상수 K_c값을 고려하여 $\cos\theta$와 $\sin\theta$를 계산하는 방법에 대해 제시하겠다. CORDIC 알고리즘에서 X와 Y의 참 값을 구하기 위해서는 상수 K_c의 값을 고려하여야 한다. 표 2의 경우에는 식 38, 39을 이용하여 풀이를 하였다. 식 38, 39을 다시 전개하면 다음과 같다.

$$\cos 15^{\circ}=\frac{X}{\sqrt{2}\cdot K_c} \tag{40}$$

$$\sin 15^{\circ}=\frac{Y}{\sqrt{2}\cdot K_c} \tag{41}$$

이때 이 수식에 K_c가 상수이기 때문에$\sqrt{2}\cdot K_c$는 또한 상수값을 갖는다. CORDIC 알고리즘은 식 35와 36은 단순히 덧셈, 뺄셈의 반복적인 식이다. 이러한 점을 고려하여 X와 Y값이 직접 $\cos\theta$와 $\sin\theta$ 값이 되도록 하기 위해서 식 35와 36에서 초기 X_0 , Y_0값을 $\sqrt{2}\cdot K_c$으로 나눈 값을 대입함으로써 X_{i+1}에서는 $\cos\theta$ 값을 구할 수 있고 Y_{i+1}에서는 $\sin\theta$ 값을 구할 수 있다.

일반적인 X와 Y값을 구하는 CORDIC 알고리즘에서는 $(\sqrt{2},\ 45^{\circ})$를 기본 벡터로 사용하고 있다. 그러나 K_c값을 고려하여 $\cos\theta$와 $\sin\theta$값을 구하는 회로 구현이 목표인 본 논문에서는 $((0.429392669\cdot\sqrt{2}),\ 45^{\circ})$을 기본 벡터로 사용한다. 본 논문의 기본 벡터를 이용하여 식 35와 36을 전개하면 표 3를 얻을 수 있다. 위와 같은 알고리즘을 이용하여 반복적인 회로 동작을 배열로 구현한다면 회로 분석이 용이할 뿐만이 아니라 속도를 개선시킬 수 있다.

기존의 반복적인 CORDIC 구조는 첫 단계의 결과 값이 다음 단계의 입력값이 되어 전체 회로에서 다시 입력으로 들어가는 구조이다. 이러한 반복형 구조가 정상동작을 하도록 제어하는 방법이 쉽지 않고, 구현된 구조에서 출력 값을 다시 입력 값으로 받기 위해서는 전체 회로 시스템이 초기의 상태를 갖도록 준비해야 한다. 이러한 구조는 회로내에서 발생하는 지연이 크고, 회로 분석이 용이하지 않다. 이러한 단점들을 보완하기 위해서 배열형 구조를 채택한다면, 속도 측면이나 회로 분석 차원에서 배열형은 다음과 같은 장점을 가지고 있다. 반복 실행시에 생길 수 있는 지연 발생이 없으며 각 단의 블럭 제어가 필요 없이 최종값을 받을 수 있어 전체적인 속도를 향상시킬 수 있다. 또한 CORDIC 회로의 동작과 구조를 이해하는데 있어서도 배열형은 분석이 용이하다. 이러한 장점을 가지고 있는 배열형 CORDIC 전체 회로도는 그림 5과 같다. 본 논문에서 설계한 회로는 최종 동작 결

〈표 3〉 벡터(X=0.429392669, Y=0.429392669)의 15°회전시 변화 단계표

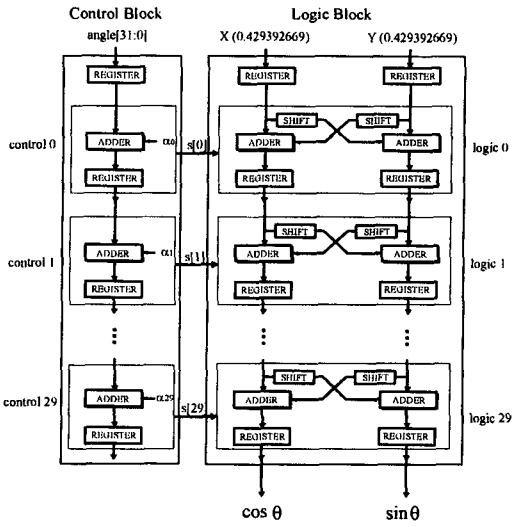
i	$\tan\alpha_i$	α_i	θ_i	ψ_i	X_{i+1}	Y_{i+1}
				−30.0000000°	0.429392669	0.429392669
0	1	−45.0000000°	−45.0000000°	+15.0000000°	0.858785338	0.000000000
1	2^{-1}	+26.5650512°	−18.4349488°	−11.5650512°	0.858785338	0.429392669
2	2^{-2}	−14.0362435°	−32.4711923°	+ 2.4711923°	0.966133505	0.214696334
3	2^{-3}	+ 7.1250163°	−25.3461760°	− 4.6538240°	0.939296463	0.335463022
4	2^{-4}	− 3.5763344°	−28.9225104°	− 1.0774896°	0.960262901	0.276756993
5	2^{-5}	− 1.7899106°	−30.7124210°	+ 0.7124210°	0.968911557	0.246748777
6	2^{-6}	+ 0.8951737°	−29.8172473°	− 0.1827527°	0.965056107	0.261888020
7	2^{-7}	− 0.4476142°	−30.2648615°	+ 0.2648615°	0.967102107	0.254348519
8	2^{-8}	+ 0.2238105°	−30.0410510°	+ 0.0410510°	0.966108558	0.258126261
9	2^{-9}	+ 0.1119057°	−29.9291453°	− 0.0708547°	0.965604405	0.260013191
10	2^{-10}	− 0.0559529°	−29.9850982°	− 0.0149018°	0.965858324	0.259070217
11	2^{-11}	− 0.0279765°	−30.0130747°	+ 0.0130747°	0.965984823	0.258598606
2	2^{-12}	+ 0.0139882°	−29.9990865°	− 0.0009135°	0.965921688	0.258834426
13	2^{-13}	− 0.0069941°	−30.0060806°	+ 0.0060806°	0.965953284	0.258716515
14	2^{-14}	+ 0.0034971°	−30.0025835°	+ 0.0025835°	0.965937493	0.258775472
15	2^{-15}	+ 00017485°	−30.0008350°	+ 0.0008350°	0.965929595	0.258804950
16	2^{-16}	+ 0.0008743°	−29.9999607°	− 0.0000393°	0.965925646	0.258819688
17	2^{-17}	− 0.0004371°	−30.0003978°	+ 0.0003978°	0.965927620	0.258812318
18	2^{-18}	+ 0.0002186°	−30.0001792°	+ 0.0001792°	0.965926632	0.258816002
19	2^{-19}	+ 0.0001093°	−30.0000699°	+ 0.0000699°	0.965926138	0.258817844
20	2^{-20}	+ 0.0000546°	−30.0000153°	+ 0.0000153°	0.965925891	0.258818765
21	2^{-21}	+ 0.0000273°	−29.0000088°	− 0.0000012°	0.965925767	0.258819225
22	2^{-22}	− 0.0000137°	−30.0000017°	+ 0.0000017°	0.965925828	0.258818994
23	2^{-23}	+ 0.0000068°	−29.0000049°	− 0.0000051°	0.965925797	0.258819109
24	2^{-24}	− 0.0000034°	−29.0000083°	− 0.0000017°	0.965925812	0.258819051
25	2^{-25}	− 0.0000017°	−30.0000000°	+ 0.0000000°	0.965925819	0.258819022
26	2^{-26}	+ 0.0000009°	−30.0000000°	+ 0.0000000°	0.965925816	0.258819036
27	2^{-27}	+ 0.0000004°	−30.0000000°	+ 0.0000000°	0.965925815	0.258819043
28	2^{-28}	+ 0.0000002°	−30.0000000°	+ 0.0000000°	0.965925814	0.258819046
29	2^{-29}	+ 0.0000001°	−30.0000000°	+ 0.0000000°	0.965925814	0.258819047

과가 $\cos\theta$와 $\sin\theta$이며 이를 위하여 기본 벡터는 $((0.429392669\cdot\sqrt{2}),\ 45^{\circ})$를 사용하였다.

3. 단계 슬라이스 형 CORDIC 파이프라인의 FPGA 구현

제안된 파이프라인 구조의 배열형 CORDIC 회로도는 그림 5과 같다. 이 회로도는 α_i의 부호 비트를 결정하는 제어부와 쉬프터, 가산기/감산기가 배열화 되어 $\cos\theta$와 $\sin\theta$값을 출력하는 연산부로 구성되어 있다. 이는 상업용 설계툴을 이용하여 구현되었다. 먼저 각 계산단계별 회로를 설계하였다. 상업용 FPGA를 사용하기 위하여 다음과 같은 사항들이 고려되었다. 먼저 칩당 수용할 수 있는 게이트 용량의 제약으로 인하여 여러칩으로 회로를 분산시켜야할 필요가 있다. 대략 60% 정도의 이용효율을 고려하면 한 개의 칩에 수용할 수 있는 회로의 크기가 결정된다.

이 회로는 총 30 단계의 파이프라인 단계로 구



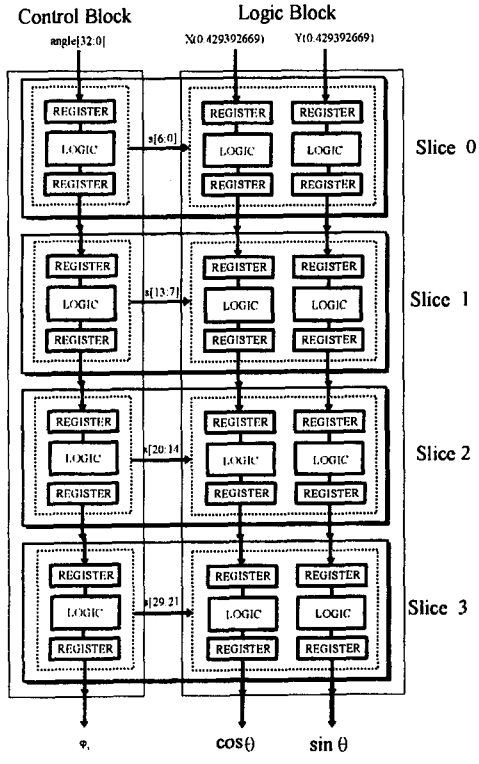
〈그림 5〉 제안된 파이프라인 구조의 배열형 CORDIC의 구조

성되며 각 단계는 대략 2000 개의 게이트로 구성되어 있다. 따라서 한 개의 칩에 수용될 수 있는 단계수가 결정된다. 이 설계에서는 칩당 6단계를 수용하여 설계되었고 소수점 이하 9자리까지의 유효숫자를 얻기위하여는 5 개의 칩을 사용해야한다.

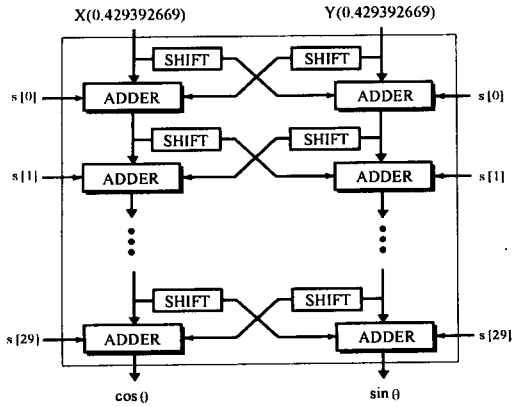
회로는 프로세서 블록과 지원회로 블록으로 분할하여 설계되었다. 프로세서 블록의 파이프라인은 단계별 슬라이스 방식으로 분할되어 각 파이프라인 슬라이스가 독립적으로 설계/제작되도록 하였다. 이 파이프라인 슬라이스를 반복시켜서 원하는 단계 수의 파이프라인 구조를 구성할 수 있도록 하였다. 그림 6에는 파이프라인형 CORDIC 회로의 구조를 보이고 있다. 이와 같은 구조는 순전히 제한된 게이트 용량의 FPGA 를 사용하기위한 방편으로 채택한 기법일 뿐이며, 단일 칩으로 설계하게 되면 이들은 단일 블록으로 합쳐질 것이다.

각 파이프라인 슬라이스는 연산부와 제어부로 구성되어 있다. 제어부는 연산을 수행해야 되는 각을 입력받아 임의의 각 α_i의 기본 벡터와의 비교에 의해서 연산부의 연산을 제어하기 위한 부호비트를 생성하여 연산부의 가산기/감산기의 선택을 결정하게된다. 제어부는 크게 두 블럭으로 나누어진다. 첫 번째로, ROM 데이터 입력블럭은 기본 벡터 45°를 고려하여 구하고자 하는 각 θ계산을 위한 것이다. 기본 벡터값 α_i을 비교기 블럭에 입력한다. 두 번째 블록으로, 비교기 블럭은 연산부에서 가산기/감산기를 결정하는$(\theta-\theta_i)$의 부호비트값을 생성한다. 비교기 블럭에서 생성된 30개의 제어신호 즉 부호비트는 연산부의 배열 회로에 각각 제어신호를 의미한다. 실제로 연산부중에서 가산기/감산기 블럭을 제어하기 위한 신호로 쓰이고 있다.

연산부는 기본벡터 0.429392669을 입력받아 제어부의 부호비트에의해 $\cos\theta$, $\sin\theta$값를 계산하여 출력하는 블럭으로 그림 7와 같다. 연산부는 식 35와 36을 계산하는 배열된 연산블럭들로 구성되어있다. 제안된 파이프라인 구조의 배열형 CORDIC 회로에서 기본 벡터로 사용된 $X_0=0.429392669$, $Y_0=0.429392669$ 은 ROM에 저장하여 첫번째 연산블럭에 인가한다. 각 연산블럭은 쌍으로 쉬프터, 가산기/감산기을 가지고 있으며 제어



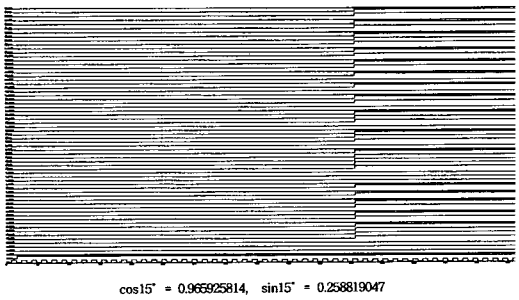
〈그림 6〉 파이프라인 형 CORDIC회로 구조



〈그림 7〉 배열형 CORDIC 회로의 연산부

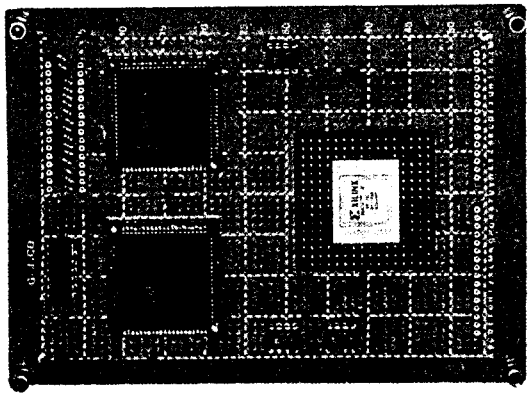
부에서 입력된 부호비트에의해 cos과 sin을 계산하게 된다. 이 블럭들은 전단에서 입력된 cos과 sin 값을 2^i만큼 값을 쉬프트 하게되고 그 값은 부호비트에 따라 덧셈 또는 뺄셈을 수행한다. 이때 쉬프트는 따로 회로를 첨가하는 것이 아니라 선의 연결만으로 쉬프트를 수행하였다. 각 연산블럭은 배열로 배열되어 연산을 수행하며 마지막 단에서 얻은 cos과 sin 직렬로 출력하게 된다. 각 연산블럭은 클럭 마다 각각의 연산을 수행하며 수행한 결과는 파이프라인 구조에 따라서 다음 연산블럭으로 전해진다.

인접한 파이프라인 슬라이스가 같은 칩내에 있지 않고 별도의 칩에 있는 경우 신호전달에 상당한 시간을 요한다. 따라서 충분한 시간의 칩간 신호전달에 할당해야하며, 이렇게 추가된 시간이 슬



$\cos 15^\circ = 0.965925814,\ \sin 15^\circ = 0.258819047$

〈그림 8〉 $\theta = 15^\circ$일때 시뮬레이션 결과



〈그림 9〉 FPGA 칩을 이용한 회로 테스트 기판 사진

라이스 내부의 파이프라인 단계의 지연시간에 영향을 미치지 않도록 할 필요가 있다. 이를 위하여 칩간 신호전송을 위하여 별도의 버퍼와 더미 슬라이스를 삽입하여 시간조절이 가능하도록 하였다. 이 더미 슬라이스에 할당된 시간은 파이프라인 한 단계의 지연시간과 같도록하여 전체적인 시스템제어의 일관성을 유지하도록 하였다. 즉 칩간 신호전달에 한 파이프라인 지연시간을 할당한 것이다. 이는 향후 단일칩화 과정에서 이 시간 간격만을 제거하여 타이밍 조절을 가능하게하는 간편함이 있다. 이렇게 설계된 파이프라인 시스템은 34 클럭이 경과하면 출력이 마지막 단에서 연속적으로 얻어지게 된다.

4. 구현결과

단계 슬라이스 형 CODIC 회로는 상업용 설계도구를 사용하여 시뮬레이션을 수행하였다. 파이프라인 단계는 16 ns에서 정상적인 동작이 확인되었다. 단계 슬라이스 구조에 대한 FPGA 칩 테스트는 별도의 인쇄회로를 제작하여 실시하였으며, 기본 클럭으로 18 MHz를 사용하였다.

그림 9는 칩 테스트를 위한 테스트 기판 사진이다. 이 테스트 보드는 클럭 발생기, 입력 신호 발생기, 타겟 칩, 디스플레이 부분으로 구분 되어있다. 클럭 발생기는 9가지의 다른 클럭을 발생시키며, 입력 신호 발생기는 타겟 칩의 입력값 즉 각도

값에 풀업 저항과 풀 다운 저항을 연결한 16단 스위치를 사용하여 입력 하도록 하였다. 디스플레이 부분은 16 개의 LED 를 2 줄로 나열하여 각각을 cos 결과값과 sin 결과 값을 2진수로 디스플레이 되도록 하였다. 타겟 칩의 결과는 16단의 LED 와 컴퓨터 연결이 가능하도록 되어 있다. 단일 입력의 결과는 LED 디스플레이를 통하여 그 결과를 볼 수 있으며, 버스트 모드에서는 컴퓨터를 통하여 그 결과값을 확인 할 수 있다.

이 단계 슬라이스형 구조에서 정밀도의 조절은 슬라이스의 개수에 의하여 조절할 수 있다. 정밀도를 낮추려면 적은 수의 슬라이스를 사용하면 된다. 이는 파이프라인 단계가 늘어나면 반복 계산의 횟수가 증가하게 되어 하위 자리의 연산이 이루어지기 때문이다. 낮은 자리의 유효숫자를 사용하지 않는 경우 유효숫자 이하는 단순히 버림으로써 계산결과를 확보할 수 있다. 따라서 입력도 상위 자리의 필요한 숫자 만큼만 사용하면 된다. 여기서 설계된 회로는 단계당 십진수 2 자리 정도의 정밀도 향상을 가져온다.

IV. 결 론

자동 항법 장치 및 위치 판독 장치의 실시간 응용이 예상되는 항공기, 유도탄 및 차량 등을 위한 고속 삼각함수 계산회로를 제안하였으며, 그 시제작을 위하여 FPGA 를 활용한 예를 제시하였다. 인공위성에서 발사되는 신호의 식별에 의하여 지는 GPS(Global Positioning System)의 측지 알고리즘이 개발되어 있으며 그 기반은 삼각함수의 계산에 있다. 기존의 삼각함수계산에 사용되는 CORDIC 알고리즘을 파이프라인 구조로 구현하여 다량의 계산을 전체적으로 신속하게 수행할 수 있는 구조를 설계하였으며, 정밀도의 요구에 따라 칩의 회로의 규모를 가변시킬 수 있도록 단계 슬라이스형 구조를 도입하였다. 이를 통하여 기존의 반복형 소프트웨어 방식보다는 현저한 속도 개선이 이루어 지게 되었다.

구현된 CORDIC회로는 마지막 두 단에서 $\cos\theta$와 $\sin\theta$값을 각각 출력할 수 있도록 하였다. 계산치와 측정치의 오차를 줄이기 위해 배열를 30단계로 하였다. 제어회로와 연산회로를 모두 파이프라인 구조를 모두 단계 슬라이스 형으로 설계하였으며, 파이프라인 슬라이스의 개수에 따라 정밀도가 달라지게 하였다. 또한 FPGA 칩을 여러개 사용하여 전체 파이프라인이 구성되는 관계로 칩간 통신에는 더미 사이클을 도입하여 칩의 입출력에 필요한 시간을 확보하는 기법을 구사하였다. 단일 칩으로 제작되는 경우 보다 빠른 계산속도와 회로의 최적화를 기할 수 있으며, 실시간 적용범위가 더욱 넓어지게 될 것이다.

참 고 문 헌

[1] Y. Bock, "Estabilshment of three-dimensional geodetic control by interferomentry with the Global Positioning System", *Journal of Geophysical Research*, vol. 90(B9), pp. 7689-7703, 1985.

[2] A. Lecik, *GPS Satellite Surveying*, A Wiley-Interscince Publication, New York, pp.233-246. 1995.

[3] J. E. Volder, "The CORDIC trigonometric computing technique", *IRE Trans. Electron. Computer.*, vol. EC-8, no. 3, pp. 330-334, Sept. 1959.

[4] D. H. Daggett, "Decimal-binary conversion in CORDIC", *IRE Trans. on Electron. Computer.*, vol. EC-8, no. 3, pp. 335-339. Sept. 1959.

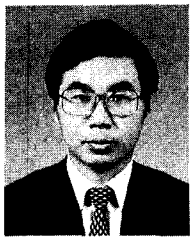
[6] N. Takagi and S. Yajima, "Redundant CORDIC methods with a constant scale factor for sine and cosine computation". *IEEE Trans. Computer*, vol. 40, no. 9, pp. 989-995, Sept. 1991.

[7] G. L. Haviland and A. A. Tuszynski, "A

CORDIC arithmetic processor chip", *IEEE Trans. Comp.*, vol C-29, no. 2, pp. 68-79. Feb. 1980

[8] V. Considine, "CORDIC trigonometric function generator for DSP", ICASSP-89, vol. 4, pp. 2381-4, May. 1989

저 자 소 개

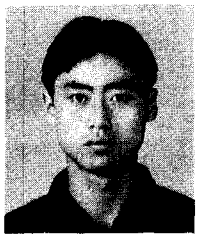


劉 泳 甲

1948年 3月 22日生
1975年 서강대학교 전자공학과 졸업(공학사)
1981年 미시간 대학교(미국) 전기전산공학과(공학석사)
1986年 미시간 대학교(미국) 전기전산공학과(공학박사)

1975年 8月～1979年 8月 국방과학연구소 연구원
1986年 2月～1988年 2月 (주) LG반도체 책임연구원
1988年 3月～ 현재 충북대학교 정보통신공학과 교수
1988年 10月～1989年 12月 (주) 한국실리콘 기술고문
1993年 1月～1994年 12月 대한전자공학회 충북지부장
1993年 8月～1994年 8月 아리조나대학교(미국) 객원교수
1994年 5月～1995年 4月 Radiance Communication, Inc., (미국) 기술고문

주관심분야 : computer architecture, memory testing, 고속시스템 설계, HDTV, ATM 가변익 항공기 제어 등



李 殷 均

1969年 2月 25日生
1994年 충북대학교 컴퓨터공학과 졸업(공학사)
1996年 충북대학교 정보통신공학과 석사과정 재학중

주관심분야 : 고속시스템 설계, VLSI 설계 등