잉여 수와 최소 반복 횟수를 이용한 CORDIC 성능 향상

Performance Enhancement of CORDIC Employing Redundant Numbers and Minimal Iterations

김승열*, 유영갑**

충북대학교 정보통신공학과*, 충북대학교 전기전자컴퓨터공학부**

Seung-Youl Kim(kimsy@hbt.chungbuk.ac.kr)*, Younggap You(ygyou@chungbuk.ac.kr)**

요약

본 논문에서는 최소 반복횟수를 갖고 잉여 수를 기반으로 하는 고성능 CORDIC 회로를 제안하였다. 최소 반복횟수는 계산미숙오차가 절단오차보다 작아지는 시점으로 결정하였다. 최소 반복 횟수는 n이 입력 각도의 비트 수 일 때 n이 $n \geq 16$이면 최소 반복 횟수는 n-4임을 알 수 있다. 이 CORDIC 회로는 잉여 수 시스템을 기반으로 변환 상수를 갖는 회로이다. 이 회로의 성능은 사인과 코사인을 계산하는데 $\{5(n-4)+2\lceil log_2 n \rceil\}\Delta$T의 지연 시간을 갖는다.

■ 중심어 : | CORDIC | 잉여 수 |

Abstract

This paper presents a high performance CORDIC circuit based on redundant numbers yielding a minimal number of iteration stages. The minimal number of iteration stages reflects the iteration number yielding a smaller computation error than the truncation error. The minimal number of iterations is found n-4 for $n \geq 16$, where n is the number of input angle bits. The CORDIC circuit is based on a redundant number system with a constant scale factor. The circuit performs sine and cosine calculations with a delay of $\{5(n-4)+2\lceil log_2 n \rceil\}\Delta$T.

■ keyword : | CORDIC | Redundant Numbers |

Ⅰ. 서 론

인터넷 및 무선통신 등의 급속한 기술 발달로 인하여 고속 마이크로프로세서, 영상처리 및 DSP (Digital Signal Processing)분야 등이 빠르게 성장하고 있다. DSP 프로세서는 복잡한 산술 연산을 빠르게 수행해야 하므로 효과적인 구현의 필요성이 증가하고 있다. 그리고 삼각함수의 계산, 좌표변환 또는 복소수 값으로 된 페이저의 계산 등은 대부분 현재의 DSP 프로세서에 포함되고 있다.

삼각함수를 계산하는 방법으로서 CORDIC(Coordinate Rotation Digital Computer) 알고리즘이 널리 사용되고

접수번호 : #060124-002
접수일자 : 2006년 01월 24일
심사완료일 : 2006년 02월 13일
교신저자 : 유영갑, e-mail : ygyou@chungbuk.ac.kr

있다[1][2]. 이 알고리즘의 속도 개선을 위한 연구가 지속적으로 수행되었다. 부호화된 잉여 수의 사용은 CORDIC 알고리즘의 속도를 향상시키기 위한 방법으로 사용되고 있다[3-5]. 잉여 수의 채택은 캐리 전파 방지 가산을 도입할 수 있도록 하여 속도를 개선하는데 중요한 역할을 한다.

잉여 수는 1 비트 이상을 사용하여 -1, 0, 1값 등을 나타낸다. 이진 수 에서는 1, 0을 이용하여 부호를 음수와 양수를 판별할 수 있다. 잉여 수는 최소 -1, 0, 1과 같이 3가지 상태를 사용함으로서 음수 및 양수 이외의 상태가 존재한다. 이진수에서 CORDIC의 크기 변환은 상수이므로 초기 값을 이용하여 보상이 가능하다. 잉여 수를 이용한 CORDIC의 크기 변환은 음수 및 양수 이외의 제3의 상태에 의하여 상수가 되지 않기 때문에 별도의 보상 회로를 필요로 하는 문제점이 있다[3][4].

잉여 수 체계를 사용하였을 때 또 하나의 중요한 문제는 그 수의 부호를 쉽게 판단할 수 없다. 모든 자리 수의 변환을 통하여 부호를 결정해야 한다. 이는 잉여 수 체계가 가지는 캐리 전파 방지의 효과를 상쇄시키는 것이다. 잉여 수를 사용한 가감산 기는 부호를 판정하기 위해 가감산 후 결과 값에 대하여 이진 가산기를 사용해야 한다. CORDIC 회로는 반복적으로 가산 또는 감산을 수행하고 반복 할 때마다 그 결과 값을 이용하여 부호를 결정하도록 되어 있다. 반복적인 가산 또는 감산 후 이진 가산기를 이용한 부호의 판별은 CORDIC 회로의 성능을 저하시킨다. 기존 2진 가산기의 캐리 전파 지연 문제가 내재하고 있다.

본 논문은 잉여 수 변환 방식을 개선하여 그 부호를 즉각 판별할 수 있는 구조를 제시한다. 잉여 수 체계가 가지는 캐리 전파 방지의 효과를 충분히 활용할 수 있도록 하였다. 그리고 부호를 즉각 판별함에 따라 잉여 수에서 발생하는 제3의 상태를 부호로 사용하지 않기 때문에 별도의 보상회로 없이 변환 상수를 사용할 수 있다. 또한 CORDIC 회로의 최소 파이프라인 단수의 결정으로 속도 향상을 가져오게 된다. CORDIC 알고리즘은 주어진 비트 수 만큼 반복 동작을 한다. 그러나 반복오차와 절단오차의 오차분석을 통하여 반복횟수를 줄이고자 한다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 기본적인 CORDIC 알고리즘을 소개하였다. Ⅲ장에서는 최소 반복 횟수를 수행하기 위한 CORDIC 파이프라인 단수의 결정에 대하여 기술하였다. Ⅳ장에서는 CORDIC 파이프라인 단의 설계에 대하여 설명하고 Ⅴ장에서는 RBSDA (Recoded Binary Signed Digit Adder) CORDIC 설계에 대하여 설명하였다. Ⅵ장에서는 설계된 CORDIC의 성능을 분석하였고 Ⅶ장에서 결론을 내렸다.

Ⅱ. CORDIC 알고리즘

삼각함수를 계산하는 CORDIC 알고리즘은 이차원 평면상의 벡터의 회전을 이용한다. CORDIC 알고리즘은 기존의 벡터 회전을 이용하여 임의 회전을 시키기 위해 다음과 같은 식으로 전개한다[1].

$$x' = x - y\tan(\phi) \tag{1}$$

$$y' = y + x\tan(\phi) \tag{2}$$

위 식의 각도 ϕ는 식(3)을 이용하여 분할하여 나타낸다.

$$\phi = \pm\alpha_0 \pm \alpha_1 \pm \alpha_2 \pm \alpha_3 \pm \alpha_4 \cdots \pm \alpha_{(i)} \tag{3}$$

분할된 각도 ϕ는 식(4)의 크기로 연속해서 회전시켜 얻는다.

$$\alpha_{(i)} = arctan 2^{-i} \tag{4}$$

구하려는 각도 z는 식 (5)와 같으며 각도는 0°로 접근시키고 $\tan(\alpha_{(i)})$는 식(6)과 같이 치환하고 $d_{(i)}$가 부호 비트일 때 다음과 같은 식으로 전개된다[3].

$$z_{(i+1)} = z_{(i)} - \alpha_{(i)} \tag{5}$$

$$\tan(\alpha_{(i)}) = d_{(i)}2^{-i} \tag{6}$$

$$x_{(i+1)} = x_{(i)} - d_{(i)}y_{(i)}2^{-i} \tag{7}$$

$$y_{(i+1)} = y_{(i)} + d_{(i)}x_{(i)}2^{-i} \tag{8}$$

$$z_{(i+1)} = z_{(i)} - d_{(i)} \tan^{-1}(2^{-i}) \quad (9)$$

Pseudo rotation에 의해 i=0,1,2,3…일 때 $x_{(i+1)}$과 $y_{(i+1)}$은 $1/\cos\alpha_{(i)}$만큼 커진다. 이 값을 보정하기 위해 i번의 반복만큼 $1/\cos\alpha_{(i)}$을 곱하여 상수 K를 구한다. 계산된 보정상수 K는 i=7일 때 약 1.6467이 된다. 초기벡터 (1, 0)인 좌표는 K 값을 보정하여 $(1/K = 0.6072, 0)$으로 나타낼 수 있다. 임의의 각 ϕ에 대하여 $\cos\phi$, $\sin\phi$는 수식(7, 8)을 이용하여 구할 수 있다. 구하려는 각도 ϕ는 식(3)과 같이 주어진 $\alpha_{(i)}$의 합으로 나타내어진다. 이때 ϕ는 $\alpha_{(i)}$값의 순서에 상관없이 구하려는 ϕ 값에 근사하면 된다. 그리고 $x_{(i+1)}$, $y_{(i+1)}$의 좌표 값은 식(4)에서 결정된 상수 $\alpha_{(i)}$의 값과 부호 d_i에 따라 변화된다. 이때 $x_{(i+1)}$, $y_{(i+1)}$좌표 값은 $\alpha_{(i)}$값의 순서에 상관없이 주어진 반복횟수 i만큼 회전하여 ϕ에 근접하였을 때 $x_{(i+1)}$, $y_{(i+1)}$의 값은 각각 $\cos\phi$, $\sin\phi$를 나타낸다.

Ⅲ. CORDIC 파이프라인 단수의 결정

최소 반복 횟수의 결정은 오차분석을 통하여 결정 할 수 있다. 최소 반복 횟수는 반복횟수 부족에 의해 발생한 계산미숙오차가 절단오차보다 작아지는 시점으로 정의한다. CORDIC 계산의 오차는 두 가지 요인에 의한다. 첫째 불충분한 반복 횟수에 의하여 생기는 계산미숙오차이다. 이는 반복 횟수를 충분히 늘려서 해결할 수 있다. 둘째 절단오차는 CORDIC 회로에서 취급하는 데이터 수에 의하여 결정된다. 오차를 줄이기 위해서는 충분한 데이터 비트 수를 사용하여야 한다.

MatLab 시뮬레이션을 이용하여 다음과 같이 오차를 분석 하였다. CORDIC 알고리즘은 데이터 비트 n에 대하여 n번 반복을 함으로서 결과 값을 얻는다. 주어진 데이터 비트 n에 대하여 매 반복 단계 마다 0°부터 360°까지 n 비트 경우의 수만큼 각도를 변화시켜 가면서 코사인과 사인 값을 계산한다. 그리고 각각의 반복 단계들 마다 계산된 코사인과 사인 값은 실제 코사인과 사인 값을 비교하여 최대 오차의 변동이 없는 단수까지 계속한다. 이 분석과정은 데이터 비트 n에 대하여 반복횟수를 증가시켜가며 계산된 코사인과 사인 값을 실제 코사인과 사인 값을 비교하여 오차를 분석한다. 그리고 최대 오차가 데이터 비트 수 n에 대하여 변화가 없을 때 반복횟수를 결정할 수 있다. 여기에서 반복횟수는 파이프라인 단 수로 나타낸다.

파이프라인 단수는 계산을 통하여 임의의 데이터 비트 수 n에 대하여 결정할 수 있다. 계산미숙 오차가 절단오차보다 작아지는 파이프라인 단수를 찾는 것이다. 이 연구에서는 데이터 비트를 8비트부터 취급하였다. 그리고 각각의 임의의 데이터 비트는 8비트씩 증가하여 48 비트까지 오차를 분석하였다.

[그림 1]은 각각의 데이터 비트에 따른 파이프라인 단수의 오차 범위를 나타내었다. 이 그림에서 입력비트 8비트부터 32비트까지는 입력 비트의 모든 경우의 수에 대하여 시뮬레이션 하였고 32비트 이상부터는 충분히 임의의 데이터를 적용하여 시뮬레이션 하였다. 오차는 각각의 데이터 비트별로 파이프라인 매 단계마다 0°에서 360°까지 계산하여 최대오차를 찾아내었다. 그림에서 보듯이 각각의 데이터 비트마다 최대 오차가 절단오차보다 작아지는 순간 반복횟수의 증가에 상관없이 오차의 변화가 없다. 이 그림에서 보듯이 8비트 데이터의 경우 7단에서 오차의 변화가 없음을 볼 수 있다. 그리고 16비트, 24비트, …, 44비트, 48비트 데이터의 경우 파이프라인 단수는 각각 12단, 20단과 같이 4단 동안 최대 오차의 변화가 거의 없음을 볼 수 있다.

파이프라인 단 수는 16 비트 이상일 때 데이터 비트 n에 대하여 n-4단으로 결정할 수 있다. [그림 1]과 같이 8bit 데이터의 경우 1단 감소한 7단으로 파이프라인 단수를 결정할 수 있다. 그리고 16bit 이상의 데이터 비트는 비트 수 n에 대하여 4단이 감소한 n-4단으로 파이프라인 단수를 결정할 수 있다.

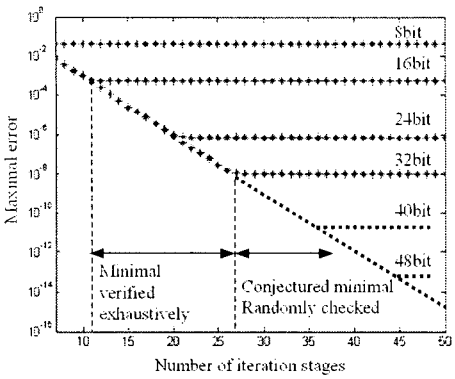

그림 1. 데이터 비트에 따른 파이프라인 단수의 오차범위

Ⅳ. CORDIC 파이프라인 단 설계

CORDIC 파이프라인 단의 고속화는 부호 결정 및 고속 가산을 통하여 개선할 수 있다. 잉여수 CORDIC의 회전 방향은 [-1,0,1]을 이용하여 결정한다. 기존의 double rotation방법은 [-1,0,1]과 같이 3가지 상태를 모두 사용한다[1]. 이 방법은 부호로서 음수 및 양수를 나타내는 [-1, 1] 뿐만 '0' 상태를 사용함으로써 크기 변환 계수는 상수가 되지 않는다. Double rotation CORDIC은 가산기를 추가하여 크기변환 계수를 상수로 사용한다[1]. 제안된 CORDIC 회로는 부호로서 [-1,1]을 사용함으로서 음수 및 양수를 제외한 제3의 상태가 존재 하지 않는다. 따라서 이진 수 CORDIC과 같이 크기 변환 계수를 상수로 사용할 수 있기 때문에 보상을 위한 추가의 회로를 필요로 하지 않는다. 이 부호의 검출은 맨체스터 캐리 체인 [7] 방식을 이용하였다.

잉여 수로 구성된 가산은 캐리 전파 문제를 해결한다. 여기에서 캐리 전파를 제거한 캐리 전파 방지 가산기는 RBSDA[6]를 사용하였다. RBSDA는 캐리 전파방지 가산을 수행하기 위해 연속된 1이나 -1이 생기지 않도록 리코딩을 수행한다. RBSD(d+, d-)을 생성하는 리코딩 블록은 가산의 결과를 입력으로 받아서 연산을 수행한다. 이 d+, d-를 생성하는 입력신호는 상호 대칭이 되도록 재구성 하였다. 그러므로 이 가산기는 캐리 전파 방지 가산을 수행하며 병렬처리를 할 수 있다. 또한 데이터 길이에 상관없이 동작시간이 일정하다.

V. RBSDA CORDIC 설계

제안된 CORDIC은 파이프라인 단을 최소화하고 캐리 전파 방지 가산을 통하여 속도를 개선할 수 있다. [그림 2]는 RBSDA CORDIC의 순서도이고 [그림 3]은 RBSDA CORDIC의 블록도이다. 이 블록은 다시 부호비트 결정블록과 회전블록으로 나뉘어져 있다.

부호 비트 결정 블록은 파이프라인 각 단계마다 하나의 레지스터와 하나의 RBSDA로 구성되어 있다. 부호비트 결정 블록은 구하려는 각도 Z, $\alpha_{(i)}$ 그리고 부호비트로 되어있다. 이때 $\alpha_{(i)}$는 이미 결정된 각도로서 파이프라인의 각 단계에 따라 다른 각도를 갖는다. 이 $\alpha_{(i)}$의 값은 이미 결정된 파이프라인의 단수에 따라 결정된다. 부호비트 결정 블록의 RBSDA 부호비트 초기 값은 음수를 사용한다. 부호비트는 RBSDA의 가산 또는 감산을 하는데 사용한다. RBSDA는 결정된 두 입력과 부호비트를 통하여 계산된다. 이때 계산된 RBSDA의 출력값은 다음 단계의 부호비트를 결정하는데 사용된다.

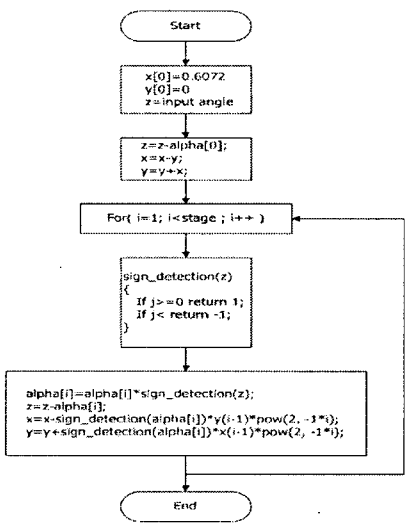

그림 2. 파이프라인 RBSDA CORDIC 순서도

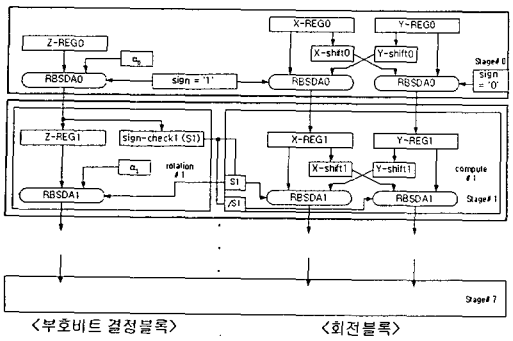

그림 3. 파이프라인 RBSDA CORDIC 구조

각 단계의 부호 비트들은 다음과 같이 결정된다. 전 단계의 RBSDA 출력에서 결정된 부호비트가 다음 파이프라인 단계의 RBSDA 부호비트의 입력으로 사용된다. 그리고 전 단계의 RBSDA의 출력 값을 다음 단계의 입력으로 사용한다. 또 다른 입력으로는 이미 결정된 $\alpha_{(i)}$의 값을 입력으로 갖는다. 이 입력을 받은 RBSDA의 출력으로부터 다음단의 부호비트를 결정한다.

RBSDA에서 부호비트의 결정은 속도에 직접적으로 영향을 미친다. 부호비트를 빠르게 결정하기 위한 방법으로 맨체스터 캐리체인[5]을 사용한다. 결정된 부호비트는 다음 단계의 부호비트 입력으로 사용된다. 결정된 각 단계의 부호비트는 회전블록의 RBSDA 부호비트 입력이 된다.

회전블록은 파이프라인 각 단계마다 2개의 레지스터, 2개의 RBSDA와 2개의 쉬프터로 구성되어 있다. 이때 쉬프터는 배선에 의한 방식으로 구성되어진다. 그리고 크게 회전블록은 x값을 계산하는 부분과 y값을 계산하는 부분으로 나누어진다. 이때 x값은 코사인 값이 되고 y값은 사인 값이 된다.

회전블록의 동작은 다음과 같다. x, y값을 계산하기 위해 RBSDA와 쉬프터가 사용된다. x값 계산을 위한 동작으로 RBSDA의 입력은 y값의 쉬프트 한 입력과 x값을 입력으로 사용한다. 마찬 가지로 y값 계산을 위한 동작으로 RBSDA의 입력은 x값의 쉬프트 한 입력과 y값을 입력으로 사용한다.

회전블록의 RBSDA의 부호 비트입력은 부호 비트 결정 블록에서 결정된 부호비트를 입력으로 받는다. 이때 x값을 계산하는 블록과 y값을 계산하는 블록의 RBSDA의 부호비트 입력은 반대이다. x값을 계산하는 블록의 RBSDA의 부호비트와 부호 비트 결정 블록의 RBSDA의 부호비트는 동일하다. 그리고 y값을 계산하는 블록의 RBSDA의 부호비트와 부호 비트 결정 블록의 RBSDA의 부호비트는 반대이다. x값을 계산하는 블록과 y값을 계산하는 블록의 RBSDA의 동작은 항상 반대이다.

설계된 8 비트 CORDIC 프로세서는 변환 상수를 갖고 잉여 수와 캐리 전파 방지 가산기를 사용하여 파이프라인 구조로 설계되었다. 설계된 CORDIC 프로세서는 회로의 동작 검증을 목적으로 VHDL을 이용하여 설계되었다. 그리고 이 회로는 Altera사의 Quartus II 2.0을 이용하여 구현 하였다. 장치는 APEX20K200EQC 240-1을 사용하였으며 전체 논리 소자의 수는 2147개이며 최대 동작주파수는 88.32MHz이다.

VI. 성능 분석

CORDIC 회로의 성능 측정은 다음과 같다. 파이프라인 한 단의 지연시간은 주로 가산기의 지연시간으로 평가하였다. 이 값들은 실제 구현 후에는 물리적 배치배선 등에 의하여 달라질 수 있다. 여기서는 상대적인 비교를 위하여 부수적인 지연요인들은 무시하였다. 그리고 하나의 게이트 지연 시간이 1ΔT라고 가정하였다.

비 잉여 수 체계를 이용한 고속 2진 가산기의 지연은 $(2\lceil log_2 n \rceil)\Delta T$이다[4]. RBSD 가산기의 지연 시간은 한 번 동작하는데 4개의 게이트를 통과하기 때문에 4 ΔT의 시간이 걸린다. 그리고 맨체스터 캐리 체인을 통하여 부호를 결정하는데 걸리는 지연 시간은 1 ΔT이다. 따라서 파이프라인 한 단의 전체지연은 데이터의 수에 상관없이 5 ΔT로 나타낼 수 있다.

제안된 RBSDA CORDIC의 지연시간은 $(5m + 2\lceil log_2 n \rceil)\Delta T$ 로 나타낼 수 있다. 여기에서 m은 파이프라인의 단 수이다. 데이터 비트가 16비트 이상일 경우 데이터 비트 n에 대하여 파이프라인 단수는 3장의 파이프라인 단수의 결정으로부터 n-4로 결정할 수 있다. 따라서 m을 n-4로 나타내면 제안된 RBSDA CORDIC의

지연시간은 $\{5(n-4)+2\lceil log_2 n\rceil\}\Delta T$로 나타낼 수 있다. 2진 가산을 이용한 CORDIC은 $(n \cdot 2\lceil log_2 n\rceil)\Delta T$ 의 지연 시간을 갖는다[4]. Double rotation방법을 이용한 CORDIC의 지연시간은 $(10n+2\lceil log_2 n\rceil)\Delta T$가 된다[4].

[표 1]은 고속 2진 가산기를 이용한 기존의 CORDIC, 잉여 수와 변환 상수를 갖는 double rotation CORDIC과 제안된 CORDIC의 지연 시간을 비교하였다. [그림 4]는 입력 비트 수에 대한 지연시간을 그래프로 나타내었다. 제안된 CORDIC 회로가 성능이 가장 우수함을 볼 수 있다.

표 2. CORDIC 회로의 지연시간 비교

	지연시간(ΔT)
기존의 CORDIC	$(n \cdot 2\lceil log_2 n\rceil)$
Double rotation CORDIC	$(10n+2\lceil log_2 n\rceil)$
제안된 CORDIC	$\{5(n-4)+2\lceil log_2 n\rceil\}$

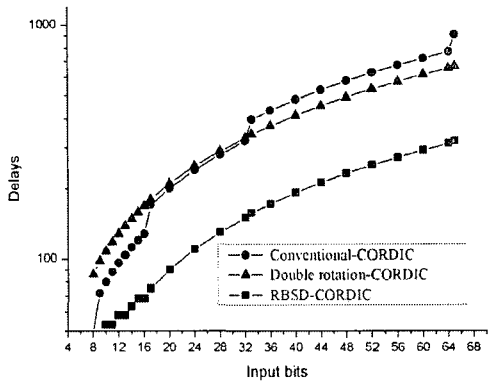

그림 4. CORDIC 지연시간 비교

Ⅶ. 결 론

최소 반복횟수와 잉여 수를 이용하여 고성능 CORDIC 회로를 제안하였다. 최소 반복횟수는 MatLab 시뮬레이션을 통하여 반복횟수 부족에 의해 발생한 계산오차가 절단오차보다 작아지는 시점으로 결정하였다. 최소 반복횟수는 파이프라인의 단수로 결정된다. 따라서 최소 파이프라인 단은 16비트 이상일 때 데이터 비트 n에 대하여 4단이 감소한 n-4단으로 구성할 수 있다. 8비트의 경우 파이프라인 단수는 1단이 감소되었다. 제안된 회로는 변환 상수와 잉여 수를 이용한 파이프라인 구조로 설계되었다. 제안된 회로는 최소 반복 횟수를 이용하여 데이터 비트 수 n이 16이상일 때 $\{5(n-4)+2\lceil log_2 n\rceil\}\Delta T$의 지연시간을 갖는다.

참 고 문 헌

[1] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electron. Computers*, Vol.EC-8, pp.330-334, Sept, 1959.

[2] J. Walther, "A unified algorithm for elementary functions," *Proc. 1971 Joint Spring Computer Conf.*, pp.379-385, July, 1971.

[3] N. Takagi, T. Asada, and S. Yajima, "Redundant CORDIC methods with a constant scale factor for sine and cosine computation," *IEEE Trans. Computers*, Vol.40, No.9, pp.989-995, Sept, 1991.

[4] T. Vladimirova and H. Tiggerler, "FPGA implementation of sine and cosine generators using the CORDIC algorithm," *Proc. MAPLD'99*, A-2, pp.28-30, Sept, 1999.

[5] M. Ercegovac and T. Lang, "Redundant and on-line CORDIC: application to matrix triangularization and SVD," *IEEE Trans. Computers*, Vol.39, pp.725-740, June, 1990.

[6] B. Parhami, "Carry-free addition of recoded binary signed-digit numbers," *IEEE Trans. Computers*, Vol.37, No.11, pp.1470-1476, Nov, 1988.

[7] T. Kilburn, D. B. G. Edwards, and D. Aspinal, "A parallel arithmetic unit using a saturated transistor fast-carry circuit," *Proc. IEE*, Vol.107 Pt. B, pp.573-584, Nov, 1960.

저 자 소 개

김 승 열(Seung-Youl Kim) 정회원

- 2002년 2월 : 충북대학교 정보통신공학과(공학사)
- 2004년 8월 : 충북대학교 정보통신공학과(공학석사)
- 2005년 3월~현재 : 충북대학교 정보통신공학과 박사과정

<관심분야> : 디지털 회로설계, Cryptography, 고속 인쇄 회로설계

유 영 갑(Younggap You) 정회원

- 1975년 8월 : 서강대학교 전자공학과(공학사)
- 1975년~1979년 : 국방과학연구소 연구원
- 1981년 8월 : Univ.of Michigan, Ann Arbor 전기전산학과(공학석사)
- 1986년 4월 : Univ.of Michigan, Ann Arbor 전기전산학과(공학박사)
- 1986년~1988년 : 금성반도체(주) 책임 연구원
- 1993년~1994년 : 아리조나 대학교 객원교수
- 1998년~2000년 : 오레곤 주립대학교 교환교수
- 1988년~현재 : 충북대학교 정보통신공학과 교수

<관심분야> : VLSI 설계 및 Test, 고속 인쇄회로 설계, Cryptography