논문 2004-41SD-12-11

AE-CORDIC: 각도 인코딩 기반 고속 CORDIC 구조

(AE-CORDIC: Angle Encoding based High Speed CORDIC Architecture)

조 용 권*, 곽 승 호*, 이 문 기**

(Yongkwon Cho, Seoungho Kwak, and Moonkey Lee)

요 약

AE-CORDIC은 CORDIC 연산의 회전 방향을 미리 계산하는 알고리즘을 이용해 CORDIC의 연산속도를 향상 시켜준다. 회전방향을 예측할 수 없는 부분은 Lookup-Table로 대체하고, 예측 가능 부분만을 CORDIC 으로 처리하였는데, 회전방향 예측은 별도의 추가 하드웨어 없이 간단하게 인코딩 할 수 있게 된다. 그리고, Unrolled CORDIC 구조에서는 Lookup-Table 입력 비트 수가 크지 않으면 Lookup-Table의 하드웨어 증가보다 CORDIC 연산 단에서 감소되는 ADDER의 하드웨어가 더 크기 때문에 오히려 전체 하드웨어 크기가 줄어든다. 본 논문에서는 회전방향 예측 가능 구간 및 예측 방법을 제안하고, 최적화된 Lookup-Table의 크기를 결정하여 기존의 회전방향 예측 알고리즘인 P-CORDIC 과 비교하였다. 그리고, 입력 각이 16비트 경우를 삼성 0.18μm 공정을 이용해 논리 합성하여 하드웨어 크기, 성능, 정확성을 검증하였다.

Abstract

AE-CORDIC improves the CORDIC operation speed with a rotation direction pre-computation algorithm. Its CORDIC iteration stages consist of non-predictable rotation direction states and predictable rotation stages. The non-predictable stages are replaced with lookup-table which has smaller hardware size than CORDIC iteration stages. The predictable stages can determine rotation direction with the input angle and simple encoder. In this paper, a rotation direction pre-computation algorithm with input angle encoder is proposed. and AE-CORDIC which have optimized Lookup-table is compared with the P-CORDIC algorithm. Hardware size, delay, and SQNR of the AE-CORDIC are verified with Samsung 0.18μm technology and Synopsys design compiler when input angle bit length is 16.

Keywords : CORDIC, 삼각함수, Trigonometric function, sin, cos

I. 서 론

삼각 함수를 계산하기 위한 CORDIC (COordinate Rotation DIgital Computer) 알고리즘은 덧셈과 쉬프트만으로 구성되어있어 하드웨어크기가 중요한 시스템에서 널리 사용되어왔다.[1] 이러한 CORDIC의 기본 연산 수식은 식(1)과 같다.

$$\begin{aligned} x_{i+1} &= x_i - \sigma_i 2^{-i} y_i \\ y_{i+1} &= y_i + \sigma_i 2^{-i} x_i \\ z_{i+1} &= z_i - \sigma_i tan^{-1}(2^{-i}) \end{aligned} \quad (1)$$

식(1)에서 σ_i 는 CORDIC의 각 반복 연산 단의 회전 방향 값을 나타낸다. 변수 z 는 연산하려는 삼각함수의 입력 각 θ 를 초기값으로 하여 각 반복 연산 단에서 회전방향에 따라 더하거나 빼주는데, 이 값이 0이 될 때까지 반복연산을 수행한다. 이러한 반복 연산 때문에 CORDIC의 처리 속도가 느려지게 된다.

CORDIC의 속도를 향상시키기 위한 다양한 방법이 연구되어왔다. RADIX-4 알고리즘은 식(1)의 2^{-i}를 기반으로 한 RADIX-2와 달리 4^{-i}를 기본으로 CORDIC 반복연산을 수행하는 방법이다. RADIX-2 방식에 비해 처리속도가 향상되지만 가변 스케일링 상수와 연산단의 복잡도 증가 문제점이 있다.[2][3] 기본적인 CORDIC 알고리즘을 그대로 유지하면서 연산 속도를 향상시키기 위한 방안으로 Redundant CORDIC이 제안되었다.[4][5] 이 방식은 덧셈의 연산 지연을 줄여주기 위한

* 학생회원, ** 정회원, 연세대학교, 전기전자공학과
(Yonsei University)
※ 본 논문은 국가지정연구실 "3차원 그래픽 가속기 개발 연구"의 지원에 의해 수행되었습니다.
접수일자: 2004년8월6일, 수정완료일: 2004년12월3일

Redundant 연산이 사용된다. 이 경우 캐리 전파(Carry Propagation)에 의한 지연은 줄여주지만 수의 표현 체계의 구조적인 문제 때문에 회전방향을 예측하기가 어렵다는 문제점이 있다.

Redundant CORDIC의 문제점을 해결하는 방법은 다음과 같이 분류할 수 있다. 첫 번째 경우는 몇몇의 비트만을 가지고 회전방향을 예측하고 보완하는 방법이다. 이중 회전 CORDIC 방식은 각 CORDIC 연산의 회전 값인 2^{-i}를 2번으로 나누어 수행하는데, 양의 방향 회전과 음의 방향 회전은 2번씩 수행해 기본적인 CORDIC 회전과 같게 수행되도록 하고, 회전방향을 예측할 수 없는 경우는 음과 양의 방향 회전을 한 번씩 수행해 회전이동을 안 한 것과 같게 만들어준다.[6] 그 결과, 회전 이동을 수행에 관계없이 일정한 스케일링 상수를 가지게 되지만, 전체적으로 연산량이 2배 증가하는 문제점을 가지고 있다. 두 번째는 차등 CORDIC 방식으로서 초기 z_0 값과 반복적 차등 연산 식을 이용해 z_i의 부호 값을 예측해낸다.[7] 이 경우, 각 연산단의 회전방향을 예측이 순차적으로 이루어지기 때문에 여전히 전체 CORDIC 연산 처리 지연 문제점이 있다. 세 번째는 미세회전(Micro-rotation)의 방향을 미리 계산하는 방법이다. 입력 각 θ 와 회전방향변수 σ 와의 관계를 모델링하여 미리 계산하는 방법(P-CORDIC)이 제안되었다.[8-10] 본 논문에서는 P-CORDIC 방식에서 사용한 회전방향 예측기와는 달리 입력 각 θ 를 통해 간단히 회전방향변수 σ 를 유도할 수 있는 방법을 제안한다. 제안하는 방법을 이용하면 σ 유도를 위한 추가 하드웨어를 사용하지 않고, 처리 지연도 발생하지 않는다.

본 논문은 Ⅱ장에서 회전방향변수 σ를 예측하는 방법을 설명하고 증명하였고, Ⅲ장에서는 제안하는 AE-CORDIC(Angle Encoding Based CORDIC) 방식의 하드웨어 크기를 비교하였다. Ⅳ장에서는 AE-CORDIC의 성능과 정확도를 살펴보고, 마지막으로 Ⅴ장에서 결론을 기술한다.

Ⅱ. 회전방향변수 σ의 예측

1. 입력 각 θ 와 회전방향변수 σ 의 관계

식 (2)는 계산하고자 하는 삼각함수의 입력 가도 변수를 보여주고, 식(3)은 CORDIC연산에서 각 단의 회전방향변수를 나타낸다.

$$\theta = \sum_{i=1}^{N} \theta_i \cdot 2^{-i} = \theta_1 \theta_2 \cdots \theta_N \tag{2}$$

$$\sigma = \sum_{i=1}^{N+1} \sigma_i \cdot 2^{-i} = \sigma_1 \sigma_2 \cdots \sigma_{N+1} \tag{3}$$

θ_i 은 입력 각의 비트 값들을 나타내고, σ_i 은 회전방향변수 σ 의 비트 값들을 나타낸다. 식(4)는 입력 각 θ 와 회전방향변수 σ_i 들의 관계를 보여준다.

$$\theta = \sigma_1 tan^{-1}(2^{-1}) + \sigma_2 tan^{-1}(2^{-2}) + \cdots + \sigma_{i-1} tan^{-1}(2^{-(i-1)}) \tag{4}$$

회전방향변수 σ_i 에 대응되는 μ_i 을 정의하면 식(5)와 같다.

$$\mu_i = \begin{cases} 1 & when\ \ \sigma_i = 1 \\ 0 & when\ \ \sigma_i = -1 \end{cases} \tag{6}$$

$$\mu = \mu_1 \mu_2 \ldots \mu_{N+1} \tag{7}$$

입력 각 θ 와 변수 μ 의 관계를 시뮬레이션을 통해 구하면 그림 1과 같다.

그림 1에서 θ 는 16비트로 LSB가 2^{-16} 을 표현하고, μ 는 17비트로서 LSB가 2^{-17} 값을 나타낸다. 입력 각은 $0 \leq \theta \leq \pi/4$ 의 범위를 가지는데, 이 범위의 값으로 다른 영역의 삼각함수 값을 구할 수 있기 때문이다. 그림 1에서 입력 각 θ 값에 변수 μ 가 부분적으로 비례한다는 것을 알 수 있다. 식(7)은 부분선형관계를 보여준다.

$$\mu = \theta + \epsilon_k \tag{8}$$

식(7)에서 k 는 θ 가 속해있는 부분선형구간을 나타내고, 상수 ε_k 는 입력 각 θ 와 회전방향 대응변수 μ 의 부분선형관계 상수 값이다. θ 가 16 비트일 경우의 부

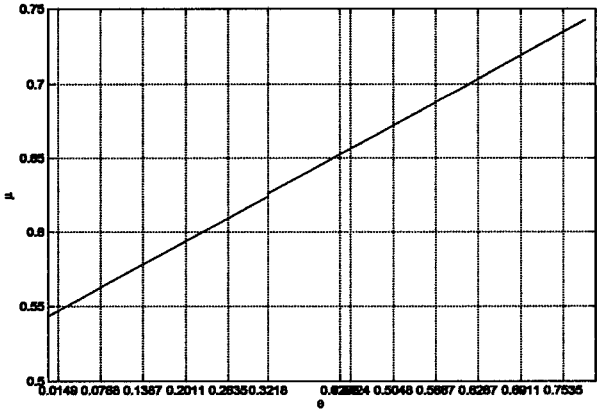

그림 1. 입력 각 θ와 변수 μ 의 부분선형관계
Fig. 1. Partially Linear Correlation Between Input Angle θ and Variable μ.

분선형구간과 상수 값은 논문[8-10]에 설명되어있다. 본 논문에서는 P-CORDIC 방식과 달리 불연속 구간은 Lookup-Table로 대체해서 처리하고, 연속 구간만 CORDIC 방식으로 처리하는 방식을 제안한다.

2. 연속 구간

두 변수가 부분선형관계를 가지는 이유는 입력 각 θ 비트들은 2^{-i} 값을 갖고 회전방향 대응변수 μ 의 비트들은 $\tan^{-1}(2^{-i})$ 값을 갖는데, i 값이 크면 선형으로 비례하지만 작으면 선형으로 비례하지 않기 때문이다. 선형 비례 i 값의 영역은 다음의 식들을 통해 구할 수 있다.

$\tan^{-1}(x)$를 테일러 시리즈로 전계하면 식(8)과 같다.[11]

$$tan^{-1}(x) = x - \frac{1}{3}x^3 + \frac{1}{5}x^5 - \cdots \quad (9)$$

식(8)에서 x 에 2^{-i} 를 대입하면 식(9)와 같다.

$$\tan^{-1}(2^{-i}) = 2^{-i} - \frac{1}{3}(2^{-i})^3 + \frac{1}{5}(2^{-i})^5 - \cdots \quad (10)$$

2^{-i} 과 $\tan^{-1}(2^{-i})$의 차이를 e_i로 정의하면 $(2^{-i})/3$보다 같거나 작아진다.

$$\begin{aligned} e_i &= 2^{-i} - tan^{-1}(2^{-1}) \\ &= 2^{-i} - 2^{-i} - \frac{1}{3}(2^{-3i}) + \frac{1}{5}(2^{-5i}) - \cdots \\ &= \frac{1}{3}(2^{-3i}) - \frac{1}{5}(2^{-5i}) + \cdots \le \frac{1}{3}(2^{-3i}) \end{aligned} \quad (11)$$

입력 각 θ 가 N 비트 이면 식(11)의 조건을 만족 시킬 경우 N 비트 표현에서 2^{-i}와 $\tan^{-1}(2^{-i})$ 가 같다고 할 수 있다.

$$e_i \le \frac{1}{3}(2^{-3i}) \le 2^{-N} \quad (11)$$

이 조건을 이용해 i 의 영역을 구하면 식(12)와 같다.

$$i \ge \frac{N - log_2 3}{3} \quad (12)$$

표 1. 은 식(12)의 결과와 시뮬레이션을 통해 불연속이 발생하지 않는 i 값을 찾은 결과를 비교하여 보여준다. 이 때, i 는 정수 값을 가지므로 계산 결과와 시뮬레이션 결과가 같아진다.

식(12)를 만족하는 최소 정수를 M 으로 정의하면, θ 가 2^{-M} 보다 작은 구간의 값인 경우 간단한 인코딩(Encoding)으로 회전방향 대응변수 μ 를 구할 수 있다. 실제로 본 논문에서는 하드웨어 추가 없이 인코딩을 구현하였다.

표 1. 불연속이 발생하지 않는 비트 수 변수 M 의 계산 결과와 시뮬레이션 결과

Table 1. Comparing Analysis and Simulation Results on Bit-length variable M for No Dis-continuous Point.

θ 비트수(N)	i 의 값	
	식(12) 계산	시뮬레이션(M)
14	4.1383	5
15	4.4717	5
16	4.8050	5
17	5.1383	6
18	5.4717	6
19	5.8050	6
20	6.1383	7
21	6.4717	7
22	6.8050	7
23	7.1383	8
24	7.4717	8
25	7.8050	8

3. 선형 비례구간에서 회전방향 예측

입력 각 θ 를 MSB 영역과 선형 비례영역으로 나누어 θ' 와 θ'' 로 표현할 수 있다.

$$\theta = \theta' + \theta'' \quad (14)$$

MSB 영역 θ' 는 2^{-1} 에서 2^{-m} 까지를 표현하고, 선형 비례영역 θ'' 는 2^{-m} 보다 작은 영역을 나타낸다. 이 때, m 은 $m \ge M$ 을 만족하는 정수 값을 가져야 한다.

선형 비례 영역 θ'' 는 식(14)와 같이 정의 할 수 있다. 이 때, θ_{m+1}, θ_{m+2}, θ_N, 들은 해당 자릿수의 비트 값을 나타낸다.

$$\begin{aligned} \theta'' &= \sum_{j=m+1}^{N} \theta_j \circ 2^{-j} \\ &= \theta_{m+1}\theta_{m+2} \ldots \theta_N \end{aligned} \quad (15)$$

입력 각 θ''는 이진(Binary) 수로 표현 되지만 회전방향변수 σ 및 대응 변수 μ 는 1과 -1의 쌍극(Bipolar) 수체계로 표현된다. 이진수 θ'' 를 쌍극 수체계로 인코딩 시키는 방법은 식(15)와 같다.

$$\begin{aligned} \theta'' &= \sum_{j=m+1}^{N} \theta_j \circ 2^{-j} \\ &= \sum_{j=m+1}^{N} [2^{-j-1} + (2\theta_j - 1) \circ 2^{-j-1}] \\ &= 2^{-m-1} - 2^{-N-1} \sum_{j=m+1}^{N} (2\theta_j - 1) \circ 2^{-j-1} \\ &= 2^{-m-1} - 2^{-N-1} \sum_{j=m+1}^{N} r_{j+1} \circ 2^{-(j+1)} \end{aligned} \quad (15)$$

이 때, 식(16)의 방법으로 θ_j 를 인코딩하여 r_{j+1}을 구한다.

$$r_{j+1} = 2\theta_j - 1 = \begin{cases} 1 & when \quad \theta_j = 1 \\ -1 & when \quad \theta_j = 0 \end{cases} \tag{16}$$

식(15)와 식(16)에서 r_j의 자리 수 값은 θ_j 보다 2^{-1} 만큼 작다는 것을 알 수 있다. 결과적으로 θ'' 을 쌍극 수 체계로 인코딩시키면 LSB 쪽에 두 자리 수가 증가한다. 이것을 자릿수 별 쌍극 값으로 나열한 것이 식(17)이다.

$$\theta'' = 1 r_{m+2} r_{m+3} \ldots r_{N+1} \bar{1} \tag{17}$$

식(17)에서 MSB와 LSB에 고정된 값은 식(15)의 마지막 줄의 상수 부분에 해당된다.

2장 2절에서 선형 비례영역에서는 2^{-i} 와 $\tan^{-1}(2^{-i})$는 동일하다는 것을 증명하였다. 이 특성을 이용하면 θ'' 의 쌍극 수로 인코딩한 식(17)을 식(4)의 회전방향변수 σ_i 로 사용할 수 있다.

$$\begin{aligned} \theta'' &= \sigma_{m+1} tan^{-1}(2^{-m-1}) + \sigma_{m+2} tan^{-1}(2^{-m-2}) + \\ &\quad \ldots + \sigma_{N+1} tan^{-1}(2^{-N-1}) + \sigma_{N+2} tan^{-1}(2^{-N-2}) \\ &= \sigma_{m+1}\sigma_{m+2} \ldots \sigma_{N+1}\sigma_{N+2} \end{aligned} \tag{19}$$

식(17)과 식(18)에서 σ_{m+1} 과 σ_{N+2} 는 상수 값을 갖기 때문에 실제 하드웨어 구현에서는 σ_{m+2} 부터 σ_{N+1} 까지만 CORDIC 연산을 수행한다.

Ⅲ. AE-CORDIC 하드웨어 구조

1. Lookup-Table

CORDIC 알고리즘을 이용한 삼각함수 연산은 입력 각 θ 를 여러 단으로 나누어서 계산한다. AE-CORDIC은 크게 Lookup-Table 부분과 CORDIC 연산 부분으로 구성된다. Lookup-Table 부분은 세 가지 역할을 수행한다. 첫 번째, 입력 각 θ 중 m 비트까지의 영역인 θ' 회전변환연산을 수행한다. $m \geq M$ 만족하면 θ 의 나머지 부분인 θ'' 영역은 선형 비례영역이 되기 때문에 회전 방향 예측 CORDIC 연산을 수행할 수 있게 된다. 둘 번째, 식(17)과 식(18)에서 입력 각 θ 의 상수 부분에 해당되는 회전 방향 σ_{m+1} 과 σ_{N+2} 의 회전변환연산을 수행한다. 항상 고정된 방향으로 회전하기 때문에 Lookup-Table에서 미리 계산해주면 CORDIC 연산 단의 하드웨어를 줄일 수 있다. 그림 2는 Lookup-Table의 첫 번째, 두 번째 역할에서 수행하는 회전변환 각을 보여준다.

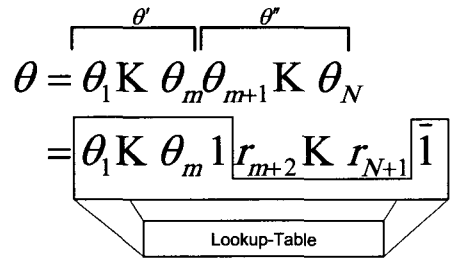

그림 2. Lookup-Table에서 수행하는 회전 변환 각
Fig. 2. The Rotation Angle of Lookup-Table.

세 번째, θ 의 나머지 부분인 θ'' 영역 CORDIC 연산 수행을 위한 배율상수(Scaling Factor) 를 처리해준다. 식(19)의 K' 는 배율상수 계산을 나타낸다.

$$K' = \sqrt{(1+2^{-2(m+2)}) \times \ldots \times (1+2^{-2(N+1)})} \tag{20}$$

Lookup-Table의 세 가지 기능을 식(20)으로 표현 할 수 있다.

$$\begin{aligned} x_{m+1} &= \frac{1}{K'}(cos\theta^T - sin\theta^T) \\ y_{m+1} &= \frac{1}{K'}(cos\theta^T + sin\theta^T) \end{aligned} \tag{21}$$

$$\theta^T = \sum_{j=1}^{m} \theta_j \cdot 2^{-j} + 2^{-(m+1)} - 2^{-(N+2)} \tag{22}$$

식(20)의 θ^T 는 Lookup-Table의 첫 번째, 두 번째 역할에서 수행하는 회전 변화의 각도로서, 식(21)과 같이 정의된다.

2. AE-CORDIC 하드웨어 구조 및 크기비교

AE-CORDIC은 Lookup-Table 부분과 CORDIC 연산 부분으로 구성된다. Lookup-Table은 CORDIC 연산 단 중 m+1 번째 까지를 대체하고 나머지 부분만 CORDIC 연산을 수행한다. 그림 3은 AE-CORDIC의 구조를 보여준다. 그림 2의 입력 각 변수 θ 의 부분 변수 θ' 는 그림 3의 Lookup-Table에 입력되고 θ'' 는 CORDIC 연산 단에서 회전방향 예측에 사용된다. 일반적인 CORDIC 방식에서는 Tan^{-1} Table 과 식(1)의 z 값 연산 부분 하드웨어가 필요하고, P-CORDIC 에서는 회전방향 계산 부분 하드웨어가 필요하다. 그렇지만,

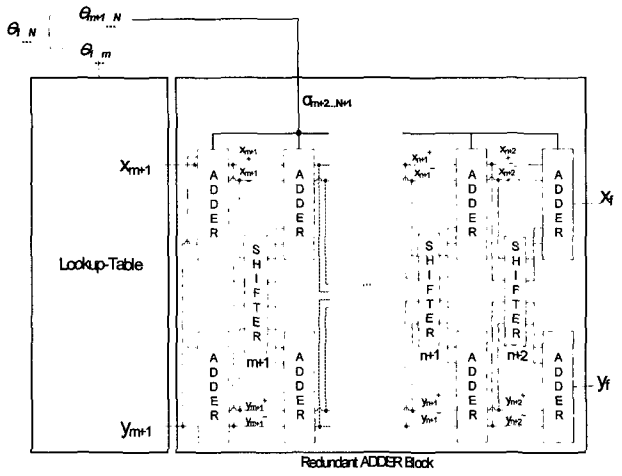

그림 3. EA-CORDIC 구조
(Lookup-Table과 Redundant ADDER 사용)
Fig. 3. EA-CORDIC Architecture.

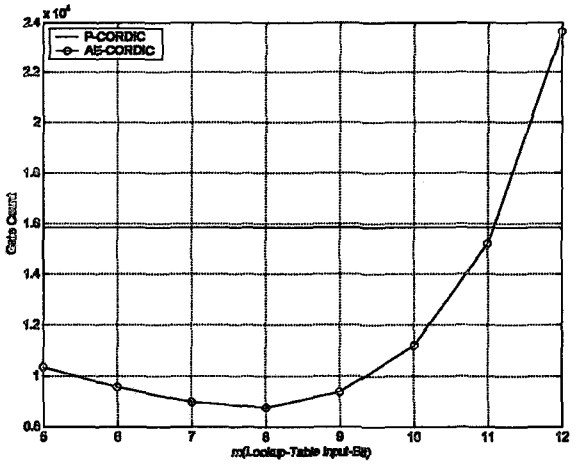

그림 4. P-CORDIC과 AE-CORDIC 게이트 카운트 비교
(θ 의 비트수 N=16, 내부 비트수=22, Unrolled 구조, Redundant 연산 사용)
Fig. 4. The Gate-count Comparison between P-CORDIC and AE-CORDIC.

AE-CORDIC 은 각도 연산이나 회전 방향 계산을 위한 하드웨어가 사용되지 않는다.

AE-CORDIC은 연산 속도를 높여주기 위해서 Re-dundant ADDER를 사용한다. 그림 3에서 Redundant ADDER 블록은 Lookup-Table의 결과 값을 받아 빠르게 CORDIC 연산을 수행한다. Redundant ADDER 블록의 연산단은 각각 Redundant ADDER 블록과 Shift 연산단으로 구성되어 있다. Redundant ADDER는 다시 일반적인 ADDER를 2개 사용한 구조로 구성되어 있고, Shift 연산단은 4개의 입력 변수를 Shift해 주는 역할을 한다. 그림 3의 Shift 연산들은 일정한 값만큼만 Shift를 하기 때문에 실제로는 하드웨어 없이 구현이 가능하다.

CORDIC 연산 단을 Lookup-Table 로 대체할 경우 CORDIC 연산 하드웨어 크기는 m 에 반비례하고 Lookup-Table의 하드웨어 크기는 2^m 에 비례해서 증가한다. 그림 4는 m 값 변화에 따른 AE-CORDIC 의 하드웨어 크기 변화와 P-CORDIC 의 하드웨어 크기 비교를 보여준다.

그림 4의 비교 하드웨어는 입력 각 θ의 비트수 N=16 이고, CORDIC 연산 단의 내부 비트수가 22며, Unrolled 구조[12]와 Redundant 연산을 채택한 것을 삼성 0.18μm 공정을 이용해 합성한 것이다. CORDIC 연산 단 크기와 Lookup-Table의 크기를 결정하는 변수 m 이 8이 될 때 까지 AE-CORDIC 하드웨어 크기가 감소하는데, 이것은 m 값이 작은 경우 Lookup-Table 증가에 따른 하드웨어 증가량보다 CORDIC 연산 단 하드웨어 크기 감소에 따른 하드웨어 감소량이 더 크기 때문이다. 이러한 관계를 이용해 최적의 하드웨어를 정할 수 있다. 그리고 m=11인 영역 까지는 여전히 AE-CORDIC의 하드웨어 크기가 작은 것을 알 수 있는데, 이것을 이용해 처리속도를 우선시 하는 최적의 구조를 얻을 수 있다.

Ⅳ. 성능 및 정밀도

AE-CORDIC은 Lookup-Table을 이용해 CORDIC 연산의 일부분은 대체하는데, Lookup-Table의 처리 지연이 대체된 CORDIC 연산 단의 처리 속도 지연보다 작기 때문에 전체 처리속도가 Lookup-Table의 크기에 따라서 향상된다. 그림 5는 Lookup-Table 크기에 따른 AE-CORDIC의 처리속도를 보여준다.

그림 5의 성능 비교는 식(12)와 표 1에 따라 θ의 비트 수가 16비트인 경우 m=5 이상일 때부터 불연속이 발생하지 않기 때문이다. 그리고, m=11 또는 12일 때 AE-CORDIC 연산지연 감소율이 둔화되는 것은 Lookup-Table 연산속도가 크기에 영향을 받기 시작했기 때문이다.

AE-CORDIC 은 Lookup-Table로 CORDIC 연산 단을 대체했기 때문에 배율상수 양자화에 의한 삼각함수 연산 정확성 감소 영향이 줄어든다. 그림 6은 Lookup-Table 크기에 따른 SQNR(Signal to Quantization Noise Ratio) 값 비교를 보여준다. 그림 6에서 AE-CORDIC의 SQNR이 P-CORDIC의 SQNR 보다 조금 향상된것을 알 수 있다. 이것은 Lookup-Table을 통

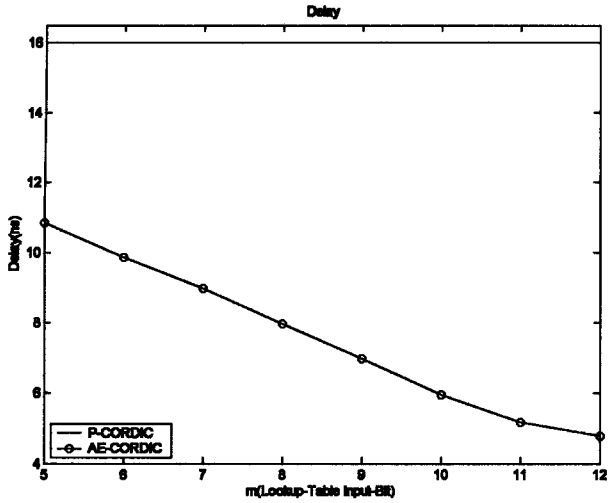

그림 5. P-CORDIC 과 AE-CORDIC의 성능 비교
Fig. 5. AE-CORDIC Performance Compared with P-CORDIC.

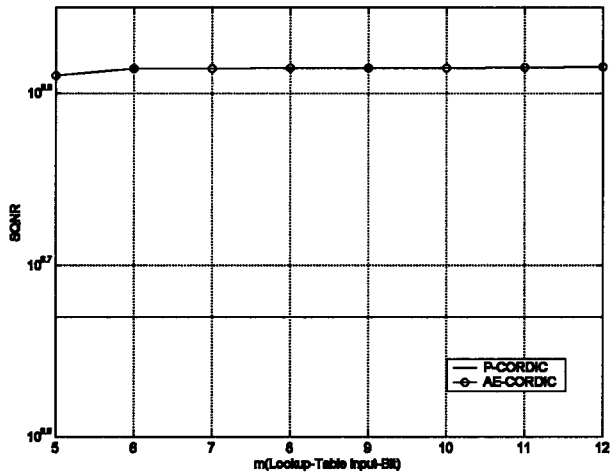

그림 6 SQNR.
Fig. 6. SQNR.

해 CORDIC 알고리즘 특성 상 발생 할 수 있는 오차까지 보정해 주었기 때문이다.

V. 결 론

P-CORDIC은 Redundant CORDIC 의 단점인 회전방향 예측하는 알고리즘이다. 본 논문에서 제안하는 AE-CORDIC 도 회전방향을 계산하는 알고리즘을 사용하는데, P-CORDIC 과 달리 MSB 영역 회전이동을 Lookup-Table로 대체하고, 나머지 부분만 CORDIC 연산하였다. 이 경우 하드웨어 추가 없이 회전방향 예측할 수 있다. 본 논문에는 AE-CORDIC의 회전방향 예측 방법을 수식적으로 증명하고, 삼성 0.18μm 공정을 이용하여 입력 각 16비트, 내부 연산 22비트 삼각함수 계산기를 게이트 수준으로 합성하였다. Unrolled CORDIC 구조에서는 AE-CORDIC을 사용할 경우 Lookup-Table의 입력 비트수가 11비트보다 작으면 P-CORDIC 보다 하드웨어 크기가 작아지면서 연산속도와 정확도가 향상된다.

참 고 문 헌

[1] J. E. Volder, "The CORDIC trigonometric computing technique," IRE Trans. Electron. Computers, vol. C-8, pp. 330-334, Sept. 1959.
[2] J. D. Bruguera, E. Antelo, and E. L. Zapata, "Design of a Pipelined Radix-4 CORDIC Processor," J. Parallel Computing, vol. 19, no. 7, pp.729-744, 1993.
[3] E. Antelo, J. Villalba, J.D. Bruguera, and E.L. Zapata, "High performance rotation architectures based on the radix-4 CORDIC algorithm," IEEE Trans. on Computers, Vol. 46, Iss. 8, pp.855-870, Aug. 1997.
[4] H. X. Lin, H. J. Sips, "On-line CORDIC algorithms, " IEEE Trans. On Computers, Vol. 39 , Iss. 8, pp. 1038-1052, Aug. 1990.
[5] 김승열, 김용대, 한선경, 유영갑, "Redundant Signed Binary Number에 의한 CORDIC 회로," 전자공학회논문지-CI, 제40권 6호, pp. 1-8, Nov. 2003.
[6] N. Takagi, T. Asada, and S. Yajima, "Redundant CORDIC methods with a constant scale factor for sine and cosine computation,"IEEE Trans. on Vol. 40, Iss. 9, pp. 989-995, Sept. 1991.
[7] H. Dawid, H. Meyr, "The Differential CORDIC Algorithm: Constant Scale Factor Redundant Implementation without Correction Iterations," IEEE Trans. on Computers. Vol. 45, no. 3, 1996.
[8] M. Kuhlmann, K.K. Parhi, "A novel CORDIC rotation method for generalized coordinate systems," Signals, Systems, and Computers Conference on ,Vol. 2, pp. 1361 - 1367, Oct. 1999.
[9] M. Kuhlmann, K.K. Parhi, "A high-speed CORDIC algorithm and architecture for DSP applications," Signal Processing Systems SiPS 99. pp. 732-741, Oct. 1999.
[10] J. Valls, M. Kuhlmann, and K.K. Parhi, "Evaluation of CORDIC Algorithms for FPGA Design,", Journal of VLSI Signal Proc., Vol. 32, Iss. 3, pp. 207-222, Nov 2002.
[11] Peter V. O'Neil, "Advanced Engineering Mathe -matics," Thomson-Engineering, 5th edition 2002.
[12] R. Andraka, "A Survey of CORDIC algorithms for FPGA based computers," ACM/SIGDA sixth

international symposium on Field Programmable Gate Arrays, pp. 192~200, Monterey, CA, Feb. 1998.

저 자 소 개

조 용 권(정회원)
1997년 연세대학교 전자공학과 학사.
2000년 연세대학교 전기컴퓨터공학과 석사.
2004년 현재 연세대학교 전기전자공학과 박사과정
<주관심분야: 마이크로프로세서, 반도체, 신호처리>

이 문 기(정회원)
1965년 연세대학교 전기공학과 학사 졸업
1967년 연세대학교 전자공학과 석사 졸업
1973년 연세대학교 전자공학과 박사 졸업
1980년 UNIV. of OKLAHOMA 전자공학 박사 졸업
1982년~현재 연세대학교 전기전자공학과 교수
<주관심분야: 마이크로프로세서, 반도체, 신호처리>

곽 승 호(정회원)
1994년 연세대학교 전자공학과 학사 졸업
1996년 연세대학교 전자공학과 석사 졸업
2004년 현재 연세대학교 전기전자공학과 박사과정
<주관심분야 : 마이크로프로세서, 반도체>