論文99-36C-3-5

# CORDIC 구조를 이용한 디지털 위상 오차 보상기의 VLSI 구현

## (VLSI Implementation of a CORDIC-based Derotator)

安 榮 虎 *, 南 勝 鉉 *, 成 元 鎔 *

(Youngho Ahn, Seunghyeon Nahm, and Wonyong Sung)

요 약

디지털 통신 시스템에서 입력 신호의 주파수와 위상 오차를 보정하는 디지털 위상 오차 보상기 (derotator)를 CORDIC (COordinate Rotation DIgital Computer) 알고리즘을 이용하여 VLSI로 구현하였다. CORDIC은 주어지는 위상값에 따라 입력 신호를 직접 회전시키므로, 디지털 주파수 합성기 (Direct Digital Frequency Synthesizer)와 복소수 승산기를 이용하는 기존의 구현 방법에 비해 하드웨어 면에서 간단하다. 디지털 위상 오차 보상기는 작은 위상 오차를 누적하므로 arctangent 함수의 선형 근사를 이용한 고속의 CORDIC 알고리즘을 이용하여 기존에 비해 약 24%의 속도 향상이 가능하였다. 본 설계된 IC는 0.6 $\mu m$ triple metal 공정을 이용하였으며, 전체 칩 면적은 6.8 $mm^2$, 트랜지스터의 개수는 11,400 개다. 측정 결과 최대 동작 주파수는 25 MHz이다.

**Abstract**

A derotator VLSI which compensates for the frequency and phase errors of a received signal in digital communication systems was developed employing a CORDIC algorithm. The CORDIC circuit directly rotates the input signal according to the phase error information, thus is much simpler than the conventional derotator architecture which consists of a DDFS (Direct Digital Frequency Synthesizer) and a complex multiplier. Since a derotator needs only small phase error accumulation, a fast direction sequence generation method which exploits the linearity of the arctangent function in small angles is utilized in order to enhance the operating speed. The chip was designed and implemented using a 0.6 $\mu m$ triple metal CMOS process by the full custom layout method. The whole chip size is 6.8 $mm^2$ and the maximum operating frequency is 25 MHz.

## I. 서 론

현재 많은 디지털 통신 시스템은 반송파 동기 등의 기능을 주파수가 낮은 기저대역 (baseband)에서 디지털 회로로 구현하기 위해, 그림 1과 같은 구조를 많이 채용하고 있다. 이 구조에서는 중간 주파수 (IF : Intermediate Frequency) 단의 복조를 위해 반송파에 동기되지 않은 발진기를 이용하고, 대신에 기저대역 (baseband)에서 디지털 위상 오차 보상기 (derotator)를 이용하여 주파수와 위상 오차를 보정한다[1]. 그림 1의 디지털 위상 오차 보상기 블록을 디지털 주파수 합성기와 복소수 승산기를 이용하여 구현하는 경우에는 주파수의 발생을 위해서 ROM과 복소 승산기를 사용해야 한다[2][3]. 한편, CORDIC 알고리즘은 여러 가지 초월 함수들을 반복 연산을 통해 계산하는 방법으로[4], 이들 중 복소수의 입력 신호를 회전시키는 기능을 이용하면 DPLL (Digital Phase Locked Loop)이나 디지털 복조기를 매우 효율적으로

* 正會員, 서울大學校 電氣工學部
(School of Electrical Engineering, Seoul National University)

接受日字:1998年8月10日, 수정완료일:1999年2月13日

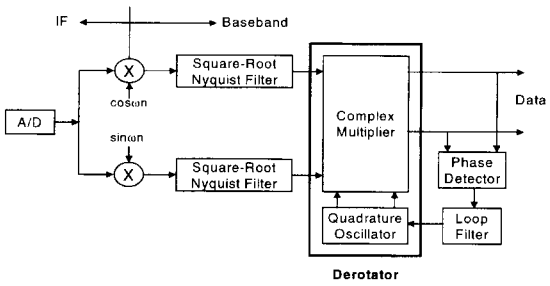



그림 1. QAM 수신기의 구조
Fig. 1. QAM receiver architecture.

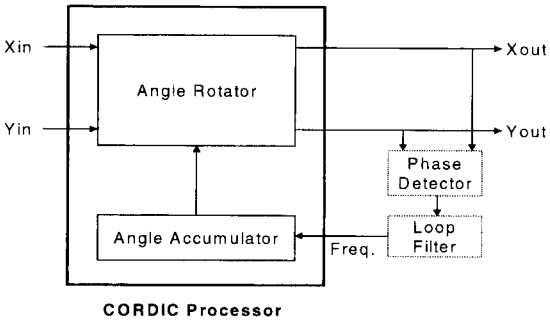



그림 2. CORDIC 프로세서를 이용한 디지털 위상 오차 보상기
Fig. 2. CORDIC-based derotator.

구현할 수 있다[5] [6]. 그림 2에서는 CORDIC을 이용하는 디지털 위상 오차 보상기의 구조를 보였다.

CORDIC 프로세서는 각계산부 (angle computation block)와 각회전부 (angle rotation block)로 구성되어 있으며, 각계산부에서는 입력된 회전 각도에 따라 각회전부를 위한 방향 제어 신호 (direct control sequence)를 발생시키고, 각회전부에서는 이 제어 신호를 바탕으로 복소수 형태의 입력 신호를 원하는 각만큼 회전한다[7]. 따라서 각계산부는 제어부 (control logic)에 해당하고 각회전부는 연산부 (data path)에 해당한다.

기존의 CORDIC 알고리즘을 병렬적으로 구현할 경우에 각계산부의 연산이 뺄셈이므로 자리올림의 전파가 매 단마다 필요하여 그 결과 매우 긴 임계경로 (critical path)를 갖는다. 또한 각계산부와 각회전부의 임계경로는 각 단에 따라 서로 달라 전체적으로는 각계산부보다 더 긴 임계경로를 갖는다. 기존에 CORDIC 알고리즘을 고속으로 구현하기 위한 방법으로는 arctangent 함수를 이용한 선형 근사 방법[8] [9], 파이프라인 구조를 사용한 방법[10], 각계산부의 방향 제어 신호를 탄젠트 함수의 선형 근사를 이용한 방법이 있다[11] [12]. Baker의 논문은 arctangent 함수의 선형성을 이용하고 앞 단의 방향 제어 신호를 ROM을 이용하여 각계산부의 임계경로를 줄이는 방식을 취하고, 각회전부에는 CSA를 사용하여 임계경로를 줄였다. 이 방식은 각회전부의 임계경로가 각계산부에 의해 거의 제한되지 않는 방식이나 각회전부와 각계산부에 보정을 위한 부분이 추가되어 하드웨어 요구량이 증가하였다. 파이프라인 구조를 이용할 경우 매우 빠른 CORDIC 프로세서를 구현할 수 있으나 그림 1 또는 2에 보이는 디지털 위상 오차 보상기와 같이 되먹임 루프 (feedback loop)가 있는 경우 시스템의 안정성 문제 때문에 이 구조가 적합하지 않다.

한편, 고속의 CORDIC을 구현하기 위해서 작은 각도에 대한 탄젠트 함수의 선형 근사를 이용하는 방법[11]은 고정된 입력에 대해서만 각 회전을 하므로 주파수 합성 등의 응용에 사용할 수 있으나 임의의 입력과 회전각을 갖는 디지털 위상 오차 보상기와 같은 경우에는 사용을 할 수 없다. 또한, 이미 주어진 회전각에 대해서 방향 제어 신호를 최적화하는 방법[12]도 회전각이 가변적인 디지털 위상 오차 보상기에는 부적합하다.

본 연구에서는 arctangent 함수의 선형 근사를 각계산부에 적용하고 각회전부는 기존의 구조를 그대로 사용을 함으로써 임의의 입력을 갖는 고속의 CORDIC 프로세서를 구현을 하였다[13].

또한 디지털 위상 오차 보상기가 작은 주파수 오차를 누적하는 점을 이용하여, 큰 각에 대한 순차적인 부분을 제거하고 탄젠트의 선형 근사만을 이용하여 하드웨어 요구량을 줄이고, 임계경로의 시간 지연을 줄였다.

한편, 각계산부의 성능 향상을 최대한 이용하기 위하여, 각회전부에 CSA (Carry Save Adder) 구조를 이용하여 임계경로를 최소화하여 각회전부의 임계경로가 각계산부의 임계경로에 거의 영향을 받지 않는 파이프라인이 없는 고속 디지털 위상 오차 보상기를 구현하였다.

본 논문의 구성은 다음과 같다. II 장에서는 디지털 위상 오차 보상기의 각계산부를 구현하기 위한 고속 알고리즘을 설명하고, III 장에서는 제안된 알고리즘을 구현하고 기존의 알고리즘과 비교하였으며, IV 장에서 측정 결과를 보이고 결론을 맺는다.

## Ⅱ. 디지털 위상 오차 보상기를 위한 고속 CORDIC 알고리즘

### 1. CORDIC 알고리즘

CORDIC 알고리즘은 여러 가지 초월 함수들의 계산뿐 아니라, 디지털 신호처리 등 여러 가지 응용에 많이 쓰이고 있다[4] [7] [10] [14] [15]. 복소수 입력 신호를 주어진 각도만큼 회전시키는 circular CORDIC 알고리즘을 간단히 기술하면 다음과 같다. 우선 직교 좌표계의 입력 신호 $[x_{in}\ y_{in}]^t$와 회전각 $z_{in}$이 주어지면 이에 의해서 초기화된다[7]. 그리고, CORDIC의 각회전부의 단수를 $n$이라고 한다면 $i=1,2,\cdots,n-1$에 대해서 다음과 같은 식이 성립한다.

$$x(i) = x(i-1) - \mu(i)y(i-1)2^{1-i} \tag{1}$$

$$y(i) = y(i-1) + \mu(i)x(i-1)2^{1-i} \tag{2}$$

$$z(i) = z(i-1) - \mu(i)a(i) \tag{3}$$

여기서 $\{\mu(i)\}$는 회전의 방향을 가리키는 부호 비트로 다음과 같이 얻어진다.

$$\mu(i) = \begin{cases} sign[z_{in}], & i=0 \\ sign[z(i-1)], & i \geq 1 \end{cases} \tag{4}$$

이 때, $a(i)$는 기본각으로 미리 정해지며, 다음과 같이 주어진다.

$$a(i) = \begin{cases} \frac{\pi}{2}, & i=0 \\ \arctan(2^{1-i}), & i \geq 1 \end{cases} \tag{5}$$

따라서 실제로 회전되는 각은 $A = \sum_{i=0}^{n-1}\mu(i)a(i)$와 같다. CORDIC 알고리즘은 식 (1)에서 식 (3)까지의 반복 연산을 수행하면서, 주어진 입력 신호를 $A$만큼 회전하는 것이 된다. 이때 식 (4)에서 구한 $\mu(i)$는 회전의 방향, +1 또는 -1,을 나타낸다. CORDIC 알고리즘이 circular 모드에서 동작하기 때문에, 각회전에 수반하여 비례상수 $K = \prod_{i=0}^{n-1}\sqrt{1+2^{-2i}}$가 곱해진 연산 결과를 얻게 되나, 디지털 위상 오차 보상기의 동작에는 영향을 주지 않는다.

CORDIC 알고리즘을 병렬구조로 구현하면 그림 3과 같다[7] [10]. 식 (1)과 (2)를 각회전부에서 수행하고, 식 (3)을 각계산부에서 처리한다. 각회전부에 필요한 ADD/SUB 블록의 제어 신호는 각계산부에 의해서 공급된다. 기존의 CORDIC 알고리즘에서는 각계산부에서 $\mu(i)$를 결정하기 위해서 $z(i)$의 부호가 필요하므로 자리올림이 매 단마다 LSB (Least Significant Bit)에서 MSB (Most Significant Bit)까지 전파되는 것을 기다려야 한다. 각계산부의 임계경로는 $a(i)$가 각 단을 거칠수록 작아지는 것을 이용하여 LSB쪽의 계산만으로 $\mu(i)$를 결정할 수 있으므로 그림 4에서 보인 것과 같이 된다. 한편 각회전부의 임계경로는 식 (1)과 (2)의 자리이동 연산 (shift operation)에 의해서 불규칙한 모양을 갖는다.

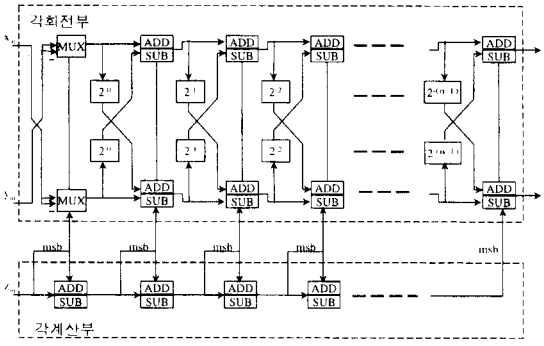



그림 3. 병렬형 CORDIC 프로세서의 구조
Fig. 3. Block diagram of the parallel CORDIC processor.

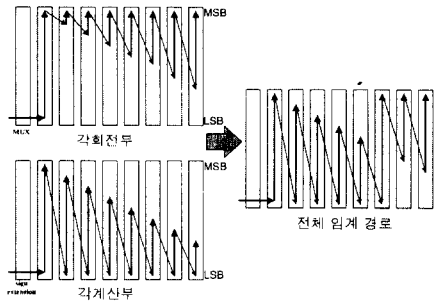



그림 4. 병렬 CORDIC 프로세서의 임계경로
Fig. 4. Critical path of the parallel CORDIC processor.

그림 4에서 각회전부의 임계경로는 회전 방향이 모두 결정되어 있다고 가정하고 임계경로를 그린 것으로, MSB 쪽의 전파를 중심으로 그렸다. 실제로는 LSB에서 MSB 쪽으로의 자리 올림의 전파도 이루어 지지만, MSB 쪽의 임계경로에 모두 포함된다. 그림 4에서 보인 바와 같이 각계산부와 각회전부의 임계경로가 서로 다른 형태를 가지면서, CORDIC 프로세서의 전체 임계경로는 더 길어진다. 이러한 까닭에 기존의

CORDIC 알고리즘은 병렬 구조에 의한 속도 증가가 매우 제한적이므로, 기존의 구현은 가산기와 자리이동 연산 (shift operation)만을 이용한 순차 구현 방식[6] 또는 파이프라인을 많이 채용한 방식이 많다[10].

2. 선형 근사화를 이용하는 고속 CORDIC 알고리즘

기존의 CORDIC 알고리즘에서 각계산부는 식 (3)과 (4)에 보이는 바와 같이, $z(i-1)$을 계산한 뒤에 그것의 부호를 구해서 다시 $z(i)$를 결정한다. 이러한 순차적인 계산에 걸리는 시간은 단 수 $n$과 입력 신호의 단어 길이에 비례한다. 입력 신호의 단어 길이는 회전의 정밀도에 해당하는 $n$에 비례하므로, 연산 시간은 대략 $n^2$에 비례한다. 이러한 문제를 해결하기 위해서 탄젠트 함수의 선형근사를 이용하여 각회전부의 임계경로를 줄이는 알고리즘을 이용한다[13]. 기본각 $a(i)$는 $i$가 커짐에 따라 작은 각도에 대해 다음과 같은 선형 근사가 가능하다.

$$2a(i)=2\arctan(2^{1-i})\approx s\cdot 2^{n-i-1} \tag{6}$$

이때, $s$는 선형 근사화를 위한 상수이며, 이러한 선형 근사가 성립되는 범위에서 여러 개의 각계산부의 값을 동시에 계산할 수 있다.

이를 좀더 자세히 설명하면 아래와 같다. 먼저 각계산부의 출력인 방향제어 $\mu(i)$를 $2b_i-1$로 치환하면 실제 회전각도 $A$는

$$\begin{aligned} A &= \frac{1}{2^n}(-a_0\cdot 2^{n-1}+\sum_{i=1}^{n-1}a_i\cdot 2^{n-i-1}) \\ &= \sum_{i=0}^{n-1}\mu(i)a(i) \\ &= \sum_{i=0}^{n-1}b_i\cdot 2a(i)-\sum_{i=0}^{n-1}a(i) \end{aligned} \tag{7}$$

가 된다. $A_0=\sum_{i=0}^{n-1}a(i)$는 상수이므로 식 (7)은 $\sum_{i=0}^{n-1}b_i\cdot 2a(i)=(A+A_0)$와 같이 나타낼 수 있다. $\mu(i)$의 정의로부터 $b_i=1$은 $+a(i)$만큼 회전하는 것을 의미하고, $b_i=0$는 $-a(i)$큼 회전하는 경우에 해당한다. 식 (6)과 (7)을 이용하여 $B=b_0b_1\ldots b_{n-1}=\sum_{i=0}^{n-1}b_i2^{n-i-1}$로 정의되는 이진수인 BSR (Binary Sequence Representation)에 의해서 $A$를 표현하면 아래와 같이 나타낼 수 있다.

$$\sum_{i=m}^{n-1}b_i\cdot 2a(i)\approx s\cdot\sum_{i=m}^{n-1}b_i\cdot 2^{n-i-1} \tag{8}$$

이때, $m$은 근사식의 성립을 보장하는 정수이다. 위의 식에서 선형성을 보장하는 $m$은 다음의 식에서 구할 수 있다[4].

$$a(m)>\sum_{i=m+1}^{n-1}a(i) \tag{9}$$

즉, 위의 식 (9)는 $a(m)$보다 작은 각들의 최대 합이 $a(m)$보다 작은 것을 의미하므로, $m$ 이하의 정수에 대해서 단조 증가가 보장된다. 이러한 $m$은 $A$의 비트 수에 따라 달라지는데 이를 모의 실험으로부터 구하면 다음의 표 1과 같다. 표 1에서 보는 것처럼, $n=9$일 경우 $m$은 3이다. 또한 식 (8)에서 선형 근사를 위한 $s$의 값은 $A$의 비트 수에 따라 표 2와 같은 관계를 갖는다. 여기서, 순차 계산이 꼭 필요한 단수, $m$는 대체적으로 전체 단수의 1/3 이하임을 알 수 있다.

표 1. 선형 근사를 위한 $m$ 값
Table 1. $m$ values for linear approximation.

| $n$ | 4 | 7 | 9 | 12 | 15 | 18 |
|---|---|---|---|---|---|---|
| $m$ | 1 | 2 | 3 | 4 | 5 | 6 |

표 2. 선형 근사를 위한 기울기 $s$ 값
Table 2. Slope for linear approximation $s$.

| $n$ | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|
| $s$ | $2\pi/52$ | $2\pi/104$ | $2\pi/204$ | $2\pi/808$ | $2\pi/1612$ | $2\pi/3228$ | $2\pi/6440$ | $2\pi/12876$ |

한편, 식 (7)의 $\sum_{i=0}^{n-1}b_i\cdot 2a(i)$을 $s$로 나누어주면, 식 (8)을 이용하여 선형성이 보장되는 부분을 BSR 형태로 나타낼 수 있다.

$$\begin{aligned} \frac{1}{s}\sum_{i=0}^{n-1}b_i2a(i) &\approx \sum_{i=0}^{m-1}b_i\cdot\frac{2a(i)}{s}+\sum_{i=m}^{n-1}b_i\cdot 2^{n-i-1} \\ &= \sum_{i=0}^{n-1}b_i\cdot 2^{n-i-1}-R(b_0b_1\ldots b_{m-1}) \\ &= B-R(b_0b_1\ldots b_{m-1}) \end{aligned} \tag{10}$$

식 (10)의 두번째 줄은 $\frac{1}{s}\sum_{i=0}^{n-1}b_i\cdot 2a(i)$를 모든 $i$에 대해서 BSR로 표현하고, 선형성이 보장되지 않는 작은 $i$에 대해서는 $R(b_0b_1\ldots b_{m-1})$를 이용하여 비선형성에 의해 생기는 오차를 보정한 것이다. 따라서 BSR $B$와 회전각 $A$는 다음과 같은 관계를 갖는다.

$$\begin{aligned} B=\sum_{i=0}^{n-1}b_i\cdot 2^{n-i-1} &= \frac{1}{s}\sum_{i=0}^{n-1}b_i2a(i)+R(b_0b_1\ldots b_{m-1}) \\ &= \frac{1}{s}(A+A_0)+R(b_0b_1\ldots b_{m-1}) \end{aligned} \tag{11}$$

만약 $n=9$인 경우 실제로 BSR과 $R(b_0b_1\ldots b_{m-1})$를

이용하여 보정된 회전 각도를 이용했을 때 $m=3$이고 $s=\frac{2\pi}{408}$가 된다. 따라서 BSR은 전체의 각도 $2\pi$를 408 단계로 나타낸다. 그림 5는 회전각 $A$와 보정되지 않은 BSR의 관계를 보인 것으로, BSR에 따라 회전각이 선형적으로 변하는 부분과 비선형적으로 변하는 부분이 있음을 알 수 있다. 보정된 BSR을 사용함으로써 BSR과 회전각도의 비선형적 관계에서 생기는 오차를 없앨 수 있다. 앞으로 본 논문에서는 별도의 언급이 없더라도 BSR이라는 용어는 보정된 BSR이란 의미로 사용한다.

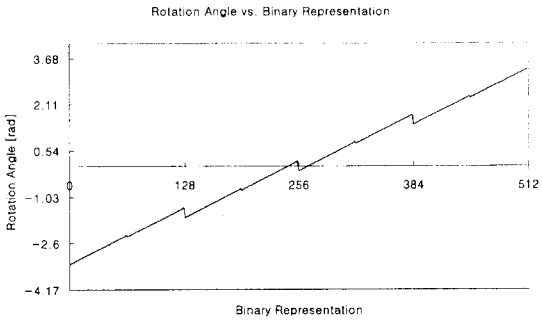



그림 5. 실제 회전각도 A와 BSR
Fig. 5. Actual rotation angle vs. BSR.

3. 누적 각회전 (Accumulated Angle Rotation)을 위한 고속 알고리즘

그림 1의 디지털 위상 오차 보상기와 같은 위상 추적 (phase tracking)의 경우에는 절대각을 회전하는 것이 아니라, 주파수 오차를 누적한 값, 즉 누적 각도 (accumulated angle)를 가지고 회전한다. 본 논문에서는 방향 제어 신호를 변형하여 방향 제어 신호를 누적하는 방식을 제안하여 각계산부의 임계경로를 줄였다. 이 때 누적 각도는 다음 식을 만족한다.

$$\begin{aligned} & A[k+1] = A[k]+\Omega[k] \\ \Leftrightarrow\ & A[k+1]+A_0 = A[k]+A_0+\Omega[k] \end{aligned} \tag{12}$$

여기서 $A[k]$는 위상이고, $\Omega[k]$는 입력으로 들어오는 주파수 오차를 나타낸다[1] [2] [3] [13]. 한편, $A_0$는 $\sum_{i=0}^{m-1} a(i)$의 값을 갖는 상수이다. 이를 BSR 영역에서 연산하기 위해서 식 (12)의 양변을 $s$로 나누어 식 (11)의 관계를 이용하면,

$$B[k+1]=B[k]+C[k]+M[k] \tag{13}$$

과 같다. 여기서 $B[k]$는 $A[k]$의, $C[k]$는 $\Omega[k]$의 BSR이 된다. 여기서 $M[k]$는 BSR을 더할 때 생기는 오차를 보정하기 위한 항으로 두 개의 BSR을 더했을 때 선형성은 보장되지 않으므로 실제 각을 더할 때와의 차이를 보정해 준다. 식 (13)과 같은 과정은 누적 각도 (accumulated angle)를 구한 뒤 BSR로 변환하는 것에 비해 하드웨어의 비용이나 속도 면에서 유리하게 된다. 식 (11)과 (13)으로부터 $M[k]$는

$$M[k]=R_B[k+1]-R_B[k]-R_C[k] \tag{14}$$

가 된다. 식 (14)에서 보면 $M[k]$는 $B[k+1]$에 의해 결정되는 $R_B[k+1]$를 알기 전에는 알 수 없다. 여기서 $B[k]$로부터 $M[k]$를 구하기 위해서는, 앞서 식 (8)의 근사화가 모든 $i$에 대해 성립하지 않는 것과 마찬가지로, $a(i)=2a(i+1)$가 모든 $i$에 대해서 성립하지 않는 점을 이용한다. 우선 식 (13)에서 $B[k]+C[k]$를 BSR로 표현하면,

$$B[k]+C[k]=\sum_{i=0}^{m-1} b_i\cdot 2^{n-i-1}+\sum_{i=0}^{m-1} c_i\cdot 2^{n-i-1} \tag{15}$$

이 때 $a(i)=2a(i+1)$가 성립하지 않을 경우는 다음과 같이 자리올림의 전파 (carry propagation)가 생기는 경우이다.

$$2^{n-i-2}+2^{n-i-2}=2^{n-i-1} \tag{16}$$

여기서 식 (16)의 좌변은 실제로 다음과 같이 쓸 수 있다.

$$\frac{2a(i+1)}{s}+\frac{2a(i+1)}{s}=\frac{2a(i)}{s}+\left[\frac{4a(i+1)}{s}-\frac{2a(i)}{s}\right] \tag{17}$$

즉 식 (17)의 우변에서 $\left[\frac{4a(i+1)}{s}-\frac{2a(i)}{s}\right]$가 BSR의 가산을 위한 보정항이 되는 것이다. 따라서 $u_i$를 $i$ 번째 위치로의 자리올림의 전파라고 한다면, 전체의 보정값은 다음과 같다.

$$M(u_1u_2\ldots u_{m-1})=\sum_{i=1}^{m-1} u_i\cdot\left[\frac{4a(i+1)}{s}-\frac{2a(i)}{s}\right] \tag{18}$$

식 (18)에서 보는 바와 같이, 보정을 위해서는 단지 $u_1\ldots u_{m-1}$에 대한 의존성 (dependency)만 있으므로 식 (3)의 과정에서 $z(i)$를 구하는 과정에 비해 매우 간단하게 된다. 예를 들어 $n=9$인 경우 $m=3$이므로 두 개의 보정항만이 필요하다.

그림 2와 같은 CORDIC 디지털 위상 오차 보상기

에 입력으로 들어오는 주파수 오차는 $\Omega[k]=\Omega_0+\Delta\Omega[k]$와 같이 나타낼 수 있다[1] [3] [13]. 여기서 $\Omega_0$는 공칭 주파수 (nominal frequency)로 디지털 위상 오차 보상기는 기저대역에서 동작하므로 0이 되고, $\Delta\Omega[k]$는 $\Omega[k]$에 비해서 매우 작은 값을 갖는 주파수 오차이다. $\Delta\Omega[k]\geq 0$일 경우에 $\Delta C[k]<2^{n-m}$이 성립하는 범위에서 $\Omega[k]$의 BSR $C[k]$는 다음과 같다.

$$C[k]=C_0+\Delta C[k]=\Delta C[k] \tag{19}$$

$\Delta\Omega[k]<0$의 경우에는 $C_0$를 달리 표현하면,

$$C[k]=C_0+\Delta C[k]=\left(\sum_{i=2}^{m-1}2^{n-i-1}+\frac{2a(1)}{s}-\sum_{i=2}^{m-1}\frac{2a(i)}{s}\right)+\Delta C[k] \tag{20}$$

가 되고, $-\frac{2a(1)}{s}+\sum_{i=2}^{m-1}\frac{2a(i)}{s}\leq\Delta C[k]$가 성립해야 한다. 따라서 디지털 위상 오차 보상기의 입력 주파수 오차는 다음과 같은 범위를 만족시켜야 한다.

$$-\frac{2a(1)}{s}+\sum_{i=2}^{m-1}\frac{2a(i)}{s}\leq\Delta C[k]<2^{n-m} \tag{21}$$

$n=9$에 대해서 $M(\cdot)$, $C_0$와 $\Delta C[k]$를 구하면 다음과 같다.

$$\begin{aligned}M(u_1) &= 18.0\approx 000010010\\ M(u_2) &= 3.78\approx 000000100\end{aligned} \tag{22}$$

$$C_0=\begin{cases}000000000, \Delta C[k]\geq 0\\ 111101010, \Delta C[k]<0\end{cases} \tag{23}$$

$$-41\leq\Delta C[k]<64 \tag{24}$$

위에서 $\Delta C[k]$는 -36.2 도와 +56.4 도에 해당한다.

## Ⅲ. CORDIC 디지털 위상 오차 보상기의 구현

본 연구에서는 $n=9$인 경우에 대해서 CORDIC 프로세서를 구현하였다. 각계산부에서는 입력 주파수 오차로부터 회전각을 나타내는 9 비트의 BSR을 생성한다. 본 논문의 제 II 장에 기술한 것처럼 입력 주파수 오차를 누적함과 동시에 이를 BSR 영역에서 보정해야 한다. 각회전부는 10 비트의 복소수 입력을 받으며 각계산부에서 들어오는 제어 신호인 BSR을 바탕으로 연산을 수행한다. 출력은 비례상수 $K$가 곱해져 있으므로 11 비트가 필요하다. 각회전부는 CSA 구조를 사용하여 지연시간을 최소화한다.

1. 누적 각계산부 (Accumulated Angle Computation Block)

누적 각계산부는 그림 6과 같다. 우선 $c_3\cdots c_{10}$은[1] 입력 주파수 오차인 $\Delta C[k]$를 나타내며, 실제 입력 부분은 실선으로 표시된 $c_3\cdots c_8$이다. BSR에 비해 입력 주파수 오차의 비트 수가 작은 것은 식 (24)에서 보인 바와 같이 주파수의 범위에 비해 주파수 오차는 작은 범위의 값을 갖기 때문이다. 점선으로 나타낸 $c_9, c_{10}$은 주파수의 정밀도를 높이기 위해서 추가가 가능한 부분이다. 이러한 입력 주파수를 누적하여, $b_0b_1\ldots b_8$로 이루어진 9 비트의 BSR을 생성한다. BSR에서 MSB는 $b_0$이고, LSB에 해당하는 것은 $b_8$이다.

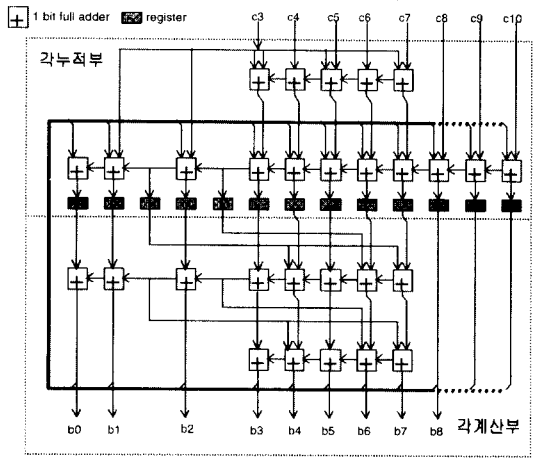



그림 6. 누적 각계산부의 블록 다이어그램
Fig. 6. Block diagram of accumulated angle computation.

실제 누적 각계산부에서 행해야 하는 계산은 식 (13)의 연산으로 다시 쓰면 아래와 같다.

$$B[k+1]=B[k]+C[k]+M[k]=B[k]+C_0+\Delta C[k]+M[k] \tag{25}$$

우선 맨 윗 단의 5 개의 전가산기 (full adder)는 $C_0+\Delta C[k]$를 연산한다. 식 (23)에서 구한 상수 $C_0$를 $\Delta C[k]$의 부호에 따라서 더하는 부분이다. 세 번째 단과 네 번째 단의 가산기는 $B[k]+M[k]$를 행하는 부분이고, 두 번째 단의 가산기에서 $(B[k]+M[k])+(C_0+\Delta C[k])$를 행한다. 이 BSR의 각 비트들이 나오

1) 입력은 $c_3$가 MSB $c_{10}$이 LSB이다.

는 순서는 $b_8b_7b_2b_6b_1b_0b_5b_4b_3$와 같다. 실제 각회전부에서는 BSR의 출력 순서를 고려하여 회전단의 순서를 맞추어 설계하였다. 그림 6에서 보이는 바와 같이 누적 각계산부에서 각계산부는 13 개의 전가산기로 구성되어 기존의 알고리즘에 비해 매우 작은 하드웨어 요구량을 가짐을 알 수 있다.

2. 각회전부 (Angle Rotation Block)

각회전부는 식 (1)와 (2)에서 보는 바와 같이 연속적인 가산과, 자리이동 연산 (shift operation)으로 이루어지며, 그림 4와 같은 임계경로를 형성한다. 자리이동 연산에 의해 각회전부의 임계경로가 길어지는 문제를 해결하기 위해서 CSA 구조를 사용한다. 일반적으로 CSA는 파이프라인된 승산기와 같이, 가산기가 병렬적으로 배열되어 있을 때 자주 사용된다[16]. 전가산기의 출력 자리올림 (output carry)을 현재 가산기의 상위 비트의 입력 자리올림 (input carry)으로 넘기지 않고, 다음 단 가산기의 자리올림으로 넘김으로써 자리올림의 계속적인 전파를 막을 수 있기 때문에 고속 알고리즘을 구현하는데 적합하다. CSA는 입력이 자리올림과 합의 형식으로 표현된 두 개의 피연산자를 가지므로, 4 비트의 입력을 받아서 2 비트의 출력을 생성한다. 한편 CORDIC의 각회전부의 경우에는 CSA 셀에서 감산이 가능해야 하므로, 이를 고려하여, CSA 셀을 설계했다.

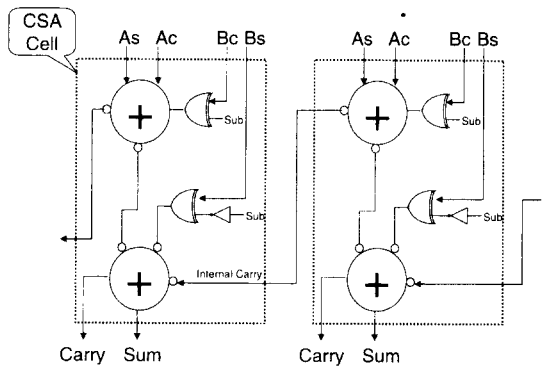



그림 7. CSA 셀의 구조
Fig. 7. Architecture of CSA cell.

그림 7은 CORDIC 프로세서에 사용된 가감산이 가능한 CSA 셀의 구조이다. $A_c, A_s$와 $B_c, B_s$는 각각의 피연산자를 자리올림과 합의 두 비트로 표현한 것이다. 2의 보수로 표현된 임의의 수 $X$를 음수로 만들기 위해서는 $\overline{X}+1$의 연산을 행해야 하므로, 피연산자 $B$를 자리올림 벡터와 합 벡터의 합으로 생각하여, 각각을 음수로 만들어야 한다. 그림 7에서 피연산자 $B_c, B_s$를 제어 신호 SUB와 배타 논리합 (exclusive OR)하고, LSB에 1을 두 개 더해야 한다. 실제로 CSA구조의 LSB 단은 입력 자리올림이 없으므로, 이들에 SUB 신호를 인가한다. 그림 7의 배타 논리합의 위치는 뒤에서 설명할 전가산기의 구조로부터 최소의 지연시간을 갖도록 배치한 것이다.

그림 7에서 보인 것처럼, 자리올림이 상위 비트로 넘어가더라도, 그 위의 비트에는 영향을 주지 않기 때문에, 자리올림의 전파가 LSB에서 MSB로 발생하지 않는다. 이러한 자리올림이 없는 성질은 CORDIC과 같이 이동연산이 많은 경우에도 매우 효율적이며, 가장 마지막 단에서 자리올림이 있는 가산기를 사용하여 자리올림과 합을 더한다.

각 계산 알고리즘을 최대한 효율적으로 이용하기 위해서 각계산부에서 먼저 나오는 BSR인 $b_8, b_7$를 이용하여, 다음 단의 BSR $b_0$가 나오기 전에 계산을 완료해야 한다. CSA 셀은 실제로 4 비트의 피연산자를 허용하는 점을 이용하여, 각계산부의 초기화를 포함한 처음 두 단을 하나의 CSA로 계산할 수 있도록 한다. 우선 식 (1)에서 (4)로 표현되는 CORDIC 알고리즘에서 $i=8,7$인 경우에 대해 전개하면,

$$
\begin{aligned}
x(0) &= x_{in} - \mu(8)2^{1-8}y_{in} \\
y(0) &= y_{in} + \mu(8)2^{1-8}x_{in} \\
x(1) &= x(0) - \mu(7)2^{1-7}y(0) \\
&= x_{in} - \mu(8)2^{1-8}y_{in} - \mu(7)2^{1-7}\{y_{in} + \mu(8)2^{1-8}x_{in}\} \\
&\simeq x_{in} - \mu(8)2^{1-8}y_{in} - \mu(7)2^{1-7}y_{in} \\
&= x_{in} - \{\mu(8)2^{1-8} + \mu(7)2^{1-7}\}y_{in} \\
y(1) &= y(0) + \mu(7)2^{1-7}x(0) \\
&= y_{in} + \mu(8)2^{1-8}x_{in} + \mu(7)2^{1-7}\{x_{in} - \mu(8)2^{1-8}y_{in}\} \\
&\simeq y_{in} + \mu(8)2^{1-8}x_{in} + \mu(7)2^{1-7}x_{in} \\
&= y_{in} + \{\mu(8)2^{1-8} + \mu(7)2^{1-7}\}x_{in}
\end{aligned}
\tag{26}
$$

와 같다. $x(1)$과 $y(1)$은 각각 $x_{in}$과 $y_{in}$의 합만으로 이루어지므로, 그림 7의 CSA 셀을 변형하여 사용하면, 각회전부의 두 단을 하나의 CSA로 구현할 수 있다. 각회전부에 대한 블록 다이어그램을 나타내면 그림 8과 같다. 이 때 각회전부가 가지는 임계경로는 CSA를 사용하였으므로 자리이동 연산에 의한 불규칙한 임계경로가 줄어든다.

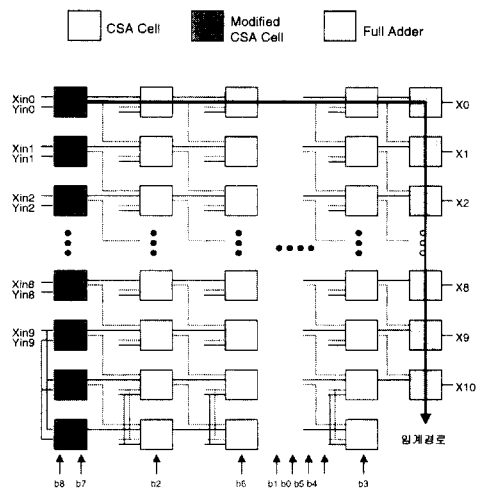



그림 8. CSA를 이용한 각회전부의 블록 다이아그램 (동상 성분)

Fig. 8. Angle rotation block using CSA. (inphase component)

3. CORDIC 프로세서의 설계

위와 같은 구조를 갖는 CORDIC 프로세서를 구현하기 위해서는 CSA 셀이 필요하며, 또한 이 CSA 셀의 변형만으로 다른 셀을 구현 할 수 있으므로 CSA 셀의 구현에 중점을 두어 설명한다. 그림 7에서 사용된 최적화된 전가산기는 $A, B, C_i$ 세 비트의 입력을 받아 다음과 같은 논리 값을 계산한다[17].

$$\begin{aligned} \overline{C_o} &= \overline{A \cdot B + C_i \cdot (A+B)} \\ \overline{S} &= \overline{A \cdot B + \overline{C_o} \cdot (A+B)} \end{aligned} \tag{27}$$

식 (17)에서 $\overline{C_o}$는 $A \cdot B$로 이루어지는 자리올림 발생 부분과 $C_i \cdot (A+B)$로 이루어지는 자리올림 전파 부분으로 이루어진다. 식 (27)로부터 $\overline{C_o}$는 1 게이트 지연시간을 갖는 것을 알 수 있다. $\overline{S}$는 합을 발생하는 회로로 역시 1 게이트 지연시간으로 구현되지만, $\overline{C_o}$가 1 게이트 지연시간을 가지므로 전가산기 전체는 2 게이트 지연시간을 갖는다. 한편, 최적화된 전가산기는 부논리 (negative logic)로도 사용될 수 있으며, 부논리의 경우에는 부논리 값인 $\overline{A}, \overline{B}, \overline{C_i}$가 들어와서 $S$와 $C_o$를 생성한다. 그림 7의 회로에서 하나의 CSA 셀은 최적화된 전가산기를 정논리 (positive logic)와 부논리를 결합하여 최적화한 구조이다. 따라서 CSA 셀 내에서 2 개의 인버터 지연시간을 줄일 수 있다.

한편, CORDIC 프로세서에서 CSA는 가감산이 가능해야 하므로, 두 개의 피연산자 중 하나를 SUB 제어 신호에 따라 음수로 변환하기 위해서 SUB 신호와 $B_c, B_s$를 배타 논리합 연산을 해야 한다. $B_c \oplus SUB$ 신호의 지연시간은 $B_c$가 $A_s$보다 1 게이트 먼저 발생되므로 전가산기의 지연시간에 흡수된다. 전단의 합인 $B_s$는 자리올림인 $B_c$보다 1 게이트 지연시간이 더 필요하므로, 그림 7의 두 번째 전가산기에 입력으로 넣어 준다. 따라서 그림 7의 CSA 셀은 약 4 게이트의 지연시간을 갖는다. 그림 9은 이러한 가감산 CSA 셀을 이용한 CORDIC 프로세서의 레이아웃 결과이다.

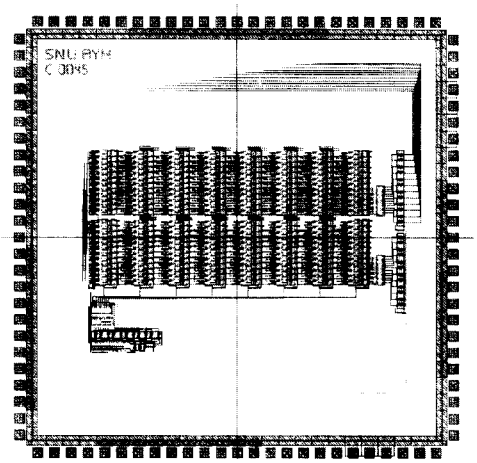


그림 9. CORDIC 구조의 디지털 위상 오차 보상기 레이아웃

Fig. 9. Layout of the CORDIC-based derotator.

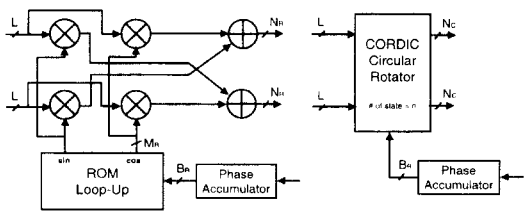
4. 구현 결과 비교

기존의 CORDIC의 임계경로는 그림 4와 같다. 식 (27)의 논리값을 갖는 최적화된 가산기를 사용할 경우, 자리올림의 경우에는 CMOS 인버터를 기준으로 1 게이트 지연시간을, 합의 경우에는 2 게이트 지연시간을 갖는다. $n=9$이고 복소수 입력이 10 비트의 정밀도를 가지는 경우를 고려하면, 우선 각회전부와 각계산부에 내에서 자리올림의 전파에 의해 각 단에서 생기는 지연시간은 $9_2+8_3+7_4+6_5+5_6+9_7+7_8+8_9=59$ 게이트 지연시간이 된다.[2] 여기에 각회전부 매 단에서는 전단에서 발생된 MSB의 합의 신호와 입력을 음수로 만들기 위한 배타 논리합 회로가 필요하므로, 매 단마다 3 게이트 지연시간이 더 필요하므로 전체 지연시간은

2) 각 숫자에 붙은 첨자는 단수를 나타낸다.

$59+3\times9=86$ 게이트 지연시간이 필요하다. 한편 Baker가 제안한 arctangent 함수의 선형성을 이용한 구조[8]에서는 본 논문에서 구현한 가감산 CSA와 유사한 구조를 사용하여 각계산부가 각회전부에 주는 영향을 극소화하였으나, 각회전부에 첫 단에 항상 회전을 하는 단과 보정의 위한 단이 추가되어 각회전부의 전체 단 수가 증가하였다. 예를 들어 $n=9$인 경우 전체 임계경로는 $(9+1)T_{CSA}+T_{CPA}+T_{ROM}=11\times4+11+10=65$가 된다.[3] 본 논문에서 구현된 그림 8의 구조는 하나의 CSA 셀이 약 4 게이트 지연시간을 갖고, $n=9$일 경우 8 개의 CSA를 지나 마지막 단의 자리올림의 전파를 갖는 가산기를 지난다. 또한 각계산부에 의해서 약 6 게이트의 지연시간이 걸리므로 총 $(9-1)T_{CSA}+T_{CPA}+T_{DELAY}=32+11+6=49$ 게이트 지연시간이 걸린다. 따라서 그림 4와 같이 기존의 병렬 알고리즘을 사용한 방법에 비해서는 40% 이상, Baker의 방식에 비하여 약 24%의 속도 향상이 가능하였다.

하드웨어 요구량을 살펴보면, 그림 6의 누적 각계산부는 입력 주파수를 누적하는 각누적부와 BSR을 이용하여 회전방향을 결정하는 각계산부로 나뉜다. 이중 각계산부는 13 개의 전가산기로만 이루어져 있다. 기존 CORDIC의 각계산부의 경우 각이 작아질수록 기본각 $a(i)$가 작아지는 것을 고려하여 그림 4의 각계산부의 임계경로와 유사하게 구현하면 약 $N+(N-1)+\cdots+(N-(n-1))=10+9+\cdots+3=52$ 개의 전가산기가 요구된다. 따라서 본 논문에서 제안한 누적 각계산부는 기존의 구조에 비해 1/3의 복잡도만을 갖는다.




(a) ROM-based (b) CORDIC-based

그림 10. ROM과 복소수 승산기를 이용한 구조 (a)와 CORDIC 구조 (b)의 하드웨어 모델

Fig. 10. Hardware model of the derotator : (a) ROM-based and (b) CORDIC-based architecture

3) Baker의 논문에서는 CSA의 임계경로를 9 게이트로 잡았으나, 알고리즘의 비교를 위해 본 논문의 CSA 구조를 이용하여 4 게이트라고 가정하였다.

한편 본 논문의 구조를 ROM과 복소 승산기를 이용한 구조와 비교하기 위해서는 ROM의 특성상 트랜지스터나 셀의 개수가 아닌 면적 비교가 필요하다. 우선 단어 길이를 결정하기 위한 하드웨어 모델은 그림 10과 같다[2][3]. 구현된 하드웨어와 동일한 해상도를 갖는 디지털 주파수 합성기를 ROM과 복소수 승산기를 사용한 경우 각 신호의 단어 길이는 $L=10$, $M_R=9$, $N_R=9$ 그리고 $B_R=10$이다. 따라서 $2^{10}\times9$ 비트의 sin/cos ROM과 입력과 ROM의 출력을 곱하기 위한 $10\times9$ 승산기가 4 개, 그리고 승산 결과를 더하기 위한 9 비트 가산기와 감산기가 필요하다. ROM의 크기는 우선 sin/cos의 대칭성을 이용할 경우 각각 $2^{(10-3)}\times9$ 비트의 ROM이 필요하고 이를 fine/coarse ROM을 이용하여 구현하면[4] $2^5\times8$ 비트의 coarse ROM와 $2^5\times3$비트의 fine ROM으로 나눌 수 있으며 fine ROM과 coarse ROM의 결과를 더하기 위한 가산기가 필요하다[18]. 하드웨어 요구량을 구하기 위한 식은 표 3에 정리한 STD90/MDL90 ASIC (Application Specific IC) 라이브러리의 하드웨어 macrocell의 면적식을 사용하였다[19].

표 3. 하드웨어 macrocell들의 면적 추정

Table 3. Area estimation of hardware macrocells.

| | |
|---|---|
| $2^N\times M$ 비트 ROM | Y = 8<br>Width = 5.8*log2( $2^N$/Y) - 120.7 + 1.6* $2^N$*Y+8.5<br>Height = 140 + 1.5* $2^N$/Y |
| $N\times M$ 파이프라인 승산기 | Width = 17.5*(N-3)<br>Height = 1.04*(111-9*M+15.5*(0.5*M-3)+123.5) |
| $N$비트 가산기 | Width = 17.5*N<br>Height = 45 |
| $N$비트 가감산기 | Width = 17.5*N<br>Height = 57 |

표 3에서 ROM은 가장 면적이 작은 DROM (Diffusion ROM)을 사용하였고, 승산기는 radix-4 modified booth 승산기를 사용하였다. CORDIC의 경우에 각 신호의 단어길이는 $L=10$, $B_c=10$, $N_c=9$ 그리고 $n=9$이다. 각회전부의 CSA를 가감산기 2 개의 하드웨어 요구량과 동일하게 해석하였으며, 전가산

4) $n$ 비트의 입력과 $W$ 비트의 출력일 때 $2^{2n/3}\times(W-1)$의 coarse ROM과 $2^{2n/3}\times\frac{W}{3}$의 fine ROM이 필요하다.

기로 구성된 각계산부는 하나의 13 비트의 가산기로 처리하였다. 표 3의 식을 이용하여 ROM과 복소 승산기를 이용한 구조의 하드웨어 요구량을 구한 것이 표 4이고, CORDIC 구조에 대해서 구한 것이 표 5이다. 표 4와 5로부터 CORDIC을 이용한 본 구조는 ROM과 승산기를 이용한 구조와 비교하여 약 14%의 하드웨어 요구량 감소가 있음을 알 수 있다.

표 4. ROM을 사용한 구조에 사용되는 하드웨어 요구량
Table 4. Hardware cost of ROM-based derotator.

| 하드웨어 | 개수 | 면적 ( $\mu m^2$ ) | 임계경로 (ns) |
|---|---|---|---|
| Coarse ROM $2^5$*8 비트 | 2 | 71014 | 2.8 |
| Fine ROM $2^5$*3 비트 | 2 | 52326 | not critical |
| 가산기 (ROM에 사용 : 9 비트) | 2 | 14175 | 1.59 |
| 승산기 (복소 승산용 : 10 비트 * 9 비트) | 4 | 246774 | 3.19 |
| 가산기 (복소 연산용 : 10 비트) | 1 | 7875 | not critical |
| 감산기 (복소 연산용 : 10 비트) | 1 | 8977 | 1.82 |
| 합 계 | | 400355 | 9.40 |

임계경로를 살펴보면, ROM과 복소수 승산기를 이용한 경우 임계경로는 $T_{ROM}+T_{RA}+T_{MUL}+T_{CPA}$이다. 이때, $T_{ROM}$은 ROM이 임계경로에서 차지하는 시간으로 $2^5\times8$ ROM의 임계경로만을 사용하고, $T_{RA}$는 fine/coarse ROM의 값을 더하기 위한 것으로 승산기에서 사용하므로 최종 MSB까지의 임계경로를 포함하였다. $T_{MUL}$의 임계경로는 상위 비트들의 결과를 더할 때 9 비트 가산기의 임계경로인 $T_{CPA}$에 그 임계경로가 포함되므로 하위 비트가 나오는 것까지만 임계경로로 계산하였다. 이를 STD90/MDL90 ASIC 라이브러리의 지연시간 식으로부터 구한 것이 표 4에 보인 바와 같이 9.4 ns이다. CORDIC의 경우 지연시간은 $(n-1)\times T_{CSA}+T_{DELAY}+T_{CPA}$가 되고, $T_{CSA}$는 2개의 전가산기 지연시간에서 2 개의 inverter 지연시간을 뺀 것으로 구하였다. $T_{DELAY}$는 각 계산부에 의한 지연으로 6 비트의 가산기와 동일한 지연시간으로 계산하였고, $T_{CPA}$는 10 비트 가산기의 지연시간이다. CORDIC을 이용한 구조의 지연시간은 표 5에서 보인 것처럼 8.45 ns이다. 따라서 임계경로 면에서 약 10 %의 성능 향상이 있었다.

표 5. CORDIC 구조에서 요구되는 하드웨어 요구량
Table 5. Hardware cost of CORDIC-based derotator.

| 하드웨어 | 개수 | 면적 ( $\mu m^2$ ) | 임계경로 (ns) |
|---|---|---|---|
| 가감산기 (각회전부 CSA : 11 비트) | 8*2*2 | 3519200 | (0.43-0.96)*2*8 |
| 가산기 (각회전부 최종단 : 11 비트) | 2 | 15750 | 1.58 |
| 가산기 (각계산부 13 비트) | 1 | 10238 | 1.32 |
| 합 계 | | 345188 | 8.45 |

## Ⅳ. 결 론

본 연구에서는 CORDIC 프로세서를 이용한 디지털 위상 오차 보상기를 구현하였다. BSR을 이용하여 위상오차 보상기의 각누적부와 CORDIC의 각계산부를 동시에 계산하는 누적 각계산부를 사용하여, 기존의 CORDIC 알고리즘와 비교하여 임계경로 면에서 약 24%의 속도 향상이 있었다. 한편, 하드웨어 요구량을 살펴보면, 각계산부의 경우 기존의 CORDIC 각계산부에 비해 1/3 정도로 줄였으며, 디지털 위상 오차 보상기는 디지털 주파수 합성기와 승산기를 이용한 구조에 비해 약 14% 정도의 하드웨어 요구량 감소와 10%의 속도 향상이 있었다. 그림 11은 CORDIC의 패턴 발생기 (pattern generator)를 이용하여 각회전부에는 상수값을 인가하고, 누적 각계산부는 주기적인 신호를 인가한 것을 논리 분석기를 이용하여 측정한 결과이다. 논리 분석기를 이용하여 구한 데이터를 DFT (Discrete Fourier Transform) 하여 주파수영역에서 관찰하면 그림 12와 같으며, 최대 spurious noise가 약 -60dB 정도인 것을 알 수 있다. 전체 칩 중 각회전부는 $3.3\times1.7$ $mm^2$의 면적을, 각계산부는 $0.8\times0.8$ $mm^2$의 면적을 차지하였으며, I/O를 제외한 전체 트랜지스터의 개수는 11,400 개다. 측정 결과 최고 동작 주파수는 25 MHz, 전력 소모는 318 mW이다.

그림 11. 주기적으로 반복되는 각 입력시의 출력 파형
Fig. 11. Waveform : periodic frequency input vector.

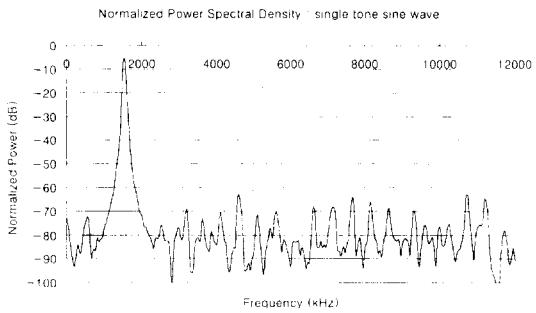



그림 12. CORDIC 프로세서 출력의 주파수 영역에서의 파형 (sine wave)

Fig. 12. Normalized power spectral density : single tone sine wave.

## 감사의 글

※ 본 연구는 교육부의 지원을 받아 반도체 공동연구소 프로젝트 95-2-2018의 일환으로 수행되었다. 또한 칩 제작 및 기술적인 지원을 해 준 LG 반도체에 감사한다.

## 참 고 문 헌

[1] B. C. Wong and H. Samueli, "A 200-MHz all-digital QAM modulator and demodulator in 1.2-$\mu$m CMOS for digital radio application," *IEEE J. Solid-State Circuits*, vol. 26, no. 12, pp. 1970-1991, Dec. 1991.

[2] H. T. Nicholas, III, and H. Samueli, "A 150-MHz direct digital frequency synthesizer in 1.25-$\mu m$ CMOS with -90dB superious performance," *IEEE J. Solid-State Circuits*, vol. 26, no. 12, pp. 1959-1969, Dec. 1991.

[3] L. K. Tan and H. Samueli, "A 200 MHz quadrature digital synthesizer/mixer in 0.8-$\mu m$ CMOS," *IEEE J. Solid-State Circuits*, vol. 30, no. 3, pp. 193-200, Mar. 1995.

[4] J. E. Volder, "A unified algorithm for elementary functions," *IRE Trans. Electron. Comput.*, vol. EC-8, pp. 330-334, Sept. 1959.

[5] A. Chen, R. McDanell, M. Boytim and R. Pogue, "Modified CORDIC demodulator implementation for digital IF-sampled receiver," *Proceedings of Globecom. '95*, vol. 2, pp. 1450-1454.

[6] J. Vuori, "Implementation of a digital phase-locked loop using CORDIC algorithm," *Proc. IEEE ISCAS '96*, vol. 4, pp. 164-168.

[7] Y. H. Hu, "CORDIC-based VLSI architectures for digital signal processing," *IEEE Signal Processing Magazine*, pp. 16-35, July 1992.

[8] P. W. Baker, "Suggestion for a fast binary sine/cosine generator," *IEEE. Trans. Computers*, vol. C-25, pp. 1134-1136, Nov. 1976.

[9] D. Timmerman, H. Hahn, and B. J. Hosticka, "Low latency time CORDIC algorithms," *IEEE. Trans. Computers*, vol. 41, no. 9, pp. 1010-1-15, Aug. 1992.

[10] G. C. Gielis, R. van de Plassche, and J. van Valburg, "A 540-MHz 10-b polar-to-cartesian converter," *IEEE J. Solid-State Circuits*, vol. 26, no. 11, pp. 1645-1650, Nov. 1991.

[11] A. Madisetti, A. Kwentus, and Jr. A. N. Willson, "A sine/cosine direct digital frequency synthesizer using an angle rotation algorithm," *ISSCC 1995 Digest of Technical Papers*, pp. 262-263, Feb 1995.

[12] Y. H. Hu and H. H.M. Chern, "A novel implementation of CORDIC algorithm using backward angle recoding (BAR)," *IEEE Trans. Computers*, pp. 1370-1378, Dec. 1996.

[13] S. Nahm and W. Sung, "A fast direction sequence generation method for CORDIC processor," *Proc. IEEE ICASSP '97*, vol. 1, pp. 635-638.

[14] J. Lee and T. Lang, "Constant-factor redundant CORDIC for angle calculation and rotation," *IEEE Trans. Computers*, vol. 41, pp. 1061-1025, Aug. 1992.

[15] M. D. Ercegovac and T. Lang, "Redundant and on-line CORDIC : applica-

tion to matrix triangularization and SVD," *IEEE Trans. Computers*, pp. 725-740, June 1990.

[16] I. Koren, *Computer Arithmetic Algorithm*, Prentice Hall, 1993.

[17] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design : a systems perspective*, MA : Addison-Wesley, 1993.

[18] D. A. Sunderland, R. A. Strauch, S. S. Wharfield, H. T. Peterson, and C. R Cole, "CMOS/SOS frequency synthesizer LSI circuit for spread spectrum communications," *IEEE J. Solid-State Circuits*, vol. Sc-19, no. 4, pp. 497-506, Aug. 1984.

[19] *STD90/MDL90 0.35 μm 3.3V CMOS standard cell library*, Samsung Electronics Co., LTD, 1998.

## 저 자 소 개

**安 榮 虎(正會員)**

1975년 1월 18일생. 1997년 2월 서울대학교 전기공학부 졸업(공학사). 1999년 2월 서울대학교 대학원 전기공학부 졸업(공학석사). 주관심분야는 디지털 신호처리, VLSI 설계

**成 元 鎔(正會員)**

1955년 4월 14일생. 1978년 2월 서울대학교 전자공학과 졸업(공학사). 1980년 2월 한국과학원 전기 및 전자공학과 졸업(공학석사). 1980년 2월 ~ 1983년 7월 금성사 중앙연구소. 1987년 7월 미국 University of California, Santa Barbara 전기 및 컴퓨터공학과 졸업(공학박사). 1989년 2월 ~ 현재 서울대학교 전기공학부 및 반도체 공동연구소 부교수. 1997년 ~ 현재 서울대학교 반도체공동연구소 집적시스템 설계센터(System EnginEering and Design center) 센터장. 주관심분야는 병렬처리 컴퓨터와 VLSI를 이용한 고속신호처리

**南 勝 鉉(正會員)**

1969년 2월 22일생. 1991년 2월 서울대학교 제어계측공학과 졸업(공학사). 1993년 2월 서울대학교 대학원 제어계측공학과 졸업(공학석사). 1998년 2월 서울대학교 대학원 전기공학부 졸업(공학박사). 1997년 10월 ~ 현재 고등기술연구원 전자통신연구실 선임연구원. 주관심분야는 디지털 신호처리, 디지털 통신 시스템의 구현