모바일 벡터 그래픽 프로세서용 역코사인 함수의 하드웨어 설계

최병윤*, 이종형**

Hardware Design of Arccosine Function for Mobile Vector Graphics Processor

Byeong-yoon Choi* · Jong-Hyoung Lee**

이 논문은 동의대학교 2008년 학술 연구비 지원에 의한 결과임

요 약

본 논문에서는 모바일 벡터 그래픽 가속기용 역코사인 연산 회로를 설계하였다. 모바일 그래픽스 응용은 기존 데스크 톱 컴퓨터에 비해 면적, 연산 시간, 전력소모와 정밀도 측면에서 제약이 크다. 설계한 역코사인 함수 회로는 연산시간과 정밀도 조건을 만족하기 위해 IEEE 표준 부동 소수점 데이터 형식을 사용하며, 계수 테이블을 사용하는 2차 다항식 근사 기법을 채택하였으며, 하드웨어 공유 기법을 통해 면적을 감소시켰다. 역코사인 회로는 약 15,280개의 게이트로 구성되며, 0.35μm CMOS 공정 조건에서 약 125 Mhz의 동작 주파수를 가진다. 7개의 클록 사이클에 역코사인 함수를 구현하므로, 설계된 회로는 약 17.85 MOPS의 연산 성능을 갖고 있어서 OpenVG 프로세서에 적용이 가능하다. 또한 융통성 있는 구조 특성으로 설계된 회로는 ROM 내용의 교체와 소규모의 하드웨어 변경을 통해 지수함수, 삼각함수, 로그 함수와 같은 다른 초월함수에 적용이 가능하다.

ABSTRACT

In this paper, the arccos($\cos^{-1}$) arithmetic unit for mobile graphics accelerator is designed. The mobile vector graphics applications need tight area, execution time, power dissipation, and accuracy constraints compared to desktop PC applications. The designed processor adopts 2nd-order polynomial approximation scheme based on IEEE floating point data format to satisfy speed and accuracy conditions and reduces area via hardware sharing structure. The arccosine processor consists of 15,280 gates and its estimated operating frequency is about 125Mhz at operating condition of 0.35μm CMOS technology. Because the processor can execute arccosine function within 7 clock cycles, it has about 17 MOPS(million arccos operations per second) execution rate and can be applicable to mobile OpenVG processor. And because of its flexible architecture, it can be applicable to the various transcendental functions such as exponential, trigonometric and logarithmic functions via replacement of ROM and minor hardware modification.

키워드

arccos 함수, OpenVG, OpenGL/ES, 2차원 그래픽스, 모바일 디바이스, SoC

* 동의대학교 컴퓨터공학과

** 동의대학교 전자공학과

접수일자 2008. 11. 10

Ⅰ. 서 론

벡터 그래픽스는 3차원 그래픽스 분야에 널리 사용되는 래스터 그래픽스와 달리 픽셀 정보가 아닌 수학식으로 이미지를 표현하므로, 이미지 확대와 축소 과정에 이미지 품질 저하가 없는 장점이 있다. 또한 동일 이미지를 표현하는데 일반적으로 래스터 그래픽스에 비해 상대적으로 적은 파일 크기가 필요하다. 이미지 크기 변환 과정에 높은 해상도를 유지하고, 전송시 작은 대역폭과 기억 공간이 필요하므로 모바일 분야에 특히 바람직하다[1,2]. 그러나 벡터 그래픽은 수학식을 처리해야 하므로 래스터 그래픽스에 비해 처리해야 할 연산양이 훨씬 크다. 따라서 벡터 그래픽스를 모바일 디바이스에 적용하기 위해서는 하드웨어 그래픽 가속기가 필요하다. 현재 그래픽 분야의 표준을 제정하고 있는 Khronous 그룹은 2차원 벡터 그래픽을 위한 공개 표준 API인 OpenVG 표준안을 정의하여 공개하고 있다[3]. OpenVG에서 2개의 연속된 세그먼트가 연결될 때, 3가지 종류의 선 결합 형식(line join style), 즉 BEVEL, ROUND, MITTER 등을 정의하고 있는데, MITTER 연결 방식을 수행하기 위해 핵심 연산으로 역코사인 연산을 필요로 한다. 현재 역코사인 삼각함수의 하드웨어 구현 연구는 다른 삼각함수, 지수함수, 대수 함수에 비해 극히 미흡한 실정이다. 컴퓨터 그래픽스 분야는 IEEE 754 부동 소수점 표준안에서 정의한 반올림 등의 정확성 조건을 필요로 하지 않으므로, 일부 오차를 허용하는 고속 연산 알고리즘이 필요하다. 일반적으로 역코사인 삼각함수를 계산하는 방법은 크게 3가지 방법으로 나뉜다. 첫 번째 방법은 CORDIC(COordinate Rotation DIgital Computer) 방식이다[4]. 이러한 방식은 반복 연산을 이용하는 구조로 연산 정확성과 구현 용이성의 장점이 있지만 많은 연산시간이 필요한 결점이 있다. 두 번째 방식은 Taylor 급수에 의한 근사 다항식을 이용하는 방식이다. 세 번째 방식은 계수 ROM 테이블을 활용하는 1차 혹은 2차 다항식 근사 방식이다[5-8]. 본 연구에서 채택한 ROM 테이블을 사용하는 다항식 근사 방식은 Taylor 급수 방식에 비해 상대적으로 낮은 차수의 다항식으로 필요한 정밀도를 제공할 수 있다. 본 논문에서는 참고문헌 [8]에서 제안한 2차 다항식 기법을 기반으로, 모바일 환경에 적합하도록 면적을 감소시키는 구조로, 부동 소수점 데이터를 입출력 데이터로 사용하는 역코사인 연산 회로를 설계하고 성능을 평가하였다.

본 논문의 구성은 다음과 같다. 제 2장에서는 역코사인 삼각함수의 특징을, 제 3 장에서는 모바일 환경에 적합한 역코사인 함수의 하드웨어 설계를 기술하였다. 제 4장에서는 설계한 회로에 대한 검증과 성능 분석을 하였으며, 마지막으로 결론을 기술하였다.

Ⅱ. 벡터그래픽스와 역코사인 함수

2.1 벡터그래픽스

컴퓨터 그래픽스 시스템은 이미지를 표현하는 방식에 따라, 그림 1과 같이 픽셀의 집합으로 표현하는 래스터 그래픽스(비트맵) 방식과 경로(path)의 집합으로 표현되는 벡터 그래픽스로 나뉜다.

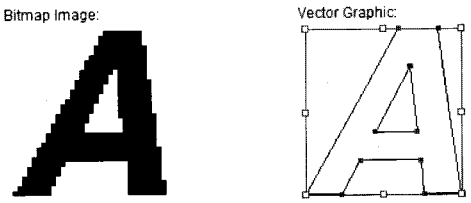

그림 1. 2가지 그래픽스 방식
(a) 래스터 그래픽스, (b) 벡터 그래픽스
Fig. 1 Two graphics schemes
(a) raster graphics, (b) vector graphics

비트맵 이미지는 개별적인 픽셀의 격자 형태로 표현되며, 벡터 그래픽스는 이미지를 기술하기 위해 점(points)과 점간을 연결하는 경로(paths)사이의 수학적인 관계를 사용한다. 이러한 벡터 그래픽스 특성으로 이미지를 확대할 경우 발생하는 앨리어싱(aliasing) 문제가 비트맵 이미지보다 훨씬 작게 된다. 이러한 2차원 이미지에 대한 벡터 그래픽스 표준안인 OpenVG는 8개의 파이프라인 단계로 이미지 데이터를 처리한다.

2.2 역코사인($\cos^{-1}$) 연산의 특징

$\cos^{-1}$ 연산은 cos 함수의 역함수로 그림 2와 같이 타당한 입력의 범위는 $-1 \leq x \leq 1$이며, 출력 범위는

$0 \leq y \leq \pi$ 이다.

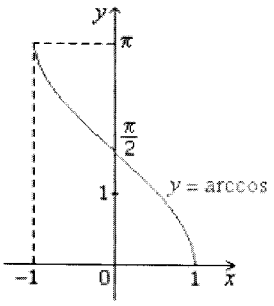

그림 2. $\cos^{-1}$ x 함수
Fig. 2 $\cos^{-1}$ x function

역코사인 계산은 x=0을 기준으로 대칭적인 특성으로 식(1)과 같이 정의될 수 있다.

$$\cos^{-1}(x) = \begin{cases} \cos^{-1}(x), & \text{if } 0 \leq x \leq 1 \\ \pi - \cos^{-1}(|x|), & \text{if } -1 \leq x < 0 \end{cases} \quad (1)$$

Ⅲ. $\cos^{-1}$ 함수의 하드웨어 설계

본 장에서는 역코사인 함수를 위한 프로세서 설계시 고려한 설계 사양과 하드웨어 설계를 기술한다.

3.1. 프로세서 설계 사양

모바일 벡터 그래픽 프로세서는 연산 성능과 함께 적절한 정밀도를 필요로 한다. 이러한 특성을 반영하여 다음과 같은 설계 사양을 정의하였다.

첫째, 연산 정밀도는 2^{-20}이상의 정밀도를 보장하도록 설계한다.

둘째, 정밀도를 향상시키기 위해 고정 소수점 형식이 아닌 단일 정밀도 부동 소수점 형식을 사용한다.

셋째, IEEE 부동 소수점 표준안에 정의한 반올림 방식과 NaN(Not-a-Number), 비정규화된 수는 지원치 않는다.

넷째, 역코사인 함수의 연산 빈도수가 덧셈, 곱셈, 나눗셈 등의 연산에 비해 적으므로, 모바일 환경의 면적 조건을 고려하여, 파이프라인 처리 구조가 아닌 반복 연산 구조를 사용한다.

3.2 역코사인 함수의 구현 알고리즘

본 논문에서는 역코사인 함수를 구현하는 방안으로 룩업 테이블 ROM과 다항식 근사를 결합한 방식을 채택하였으며, 그림 3과 같이 크게 3 단계로 나누어진다.

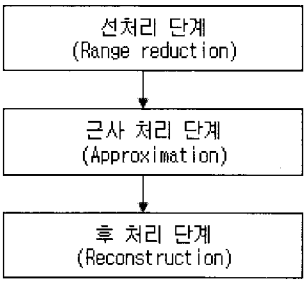

그림 3. 3 단계 처리
Fig. 3 Three-step operation

첫 번째 단계는 근사에 필요한 다항식 수를 최소화하고 계산을 효율적으로 수행하기 위해, 입력 인수의 범위를 감소시키는 단계이다. 두 번째 단계는 다항식 근사를 수행하는 과정이다. 그리고 마지막 단계는 첫 번째 단계에서 수행한 범위 제한 조건을 보정하여 원하는 결과를 생성하는 과정이다. 기존의 룩업 테이블을 사용하는 근사 방식은 타당한 입력 구간을 세부 구간 $h = x_{i+1} - x_i$ 단위로 균등하게 나누고, 세부 구간의 양 끝점에 대한 함수의 정확한 값을 계산해서 룩업 테이블에 저장한 후, 저장된 표본점에 대응하는 입력 데이터의 상위 비트를 사용하여, 그림 4와 같이 테이블 룩업에 의해 입력 데이터에 대한 근사 값을 고속으로 구할 수 있다. 이 방식은 입력 데이터의 상위 비트를 ROM 테이블의 입력으로 세부 구간이 경계에서는 정확한 값을 구할 수 있지만, 세부 구간의 중간 부분에 대해서는 오차가 크게 된다. 따라서 세부 구간의 중간 영역의 입력 값에 대해 보다 정확한 근사 값을 계산하기 1 차 혹은 2차 근사 다항식을 사용하는 것이 일반적이다.

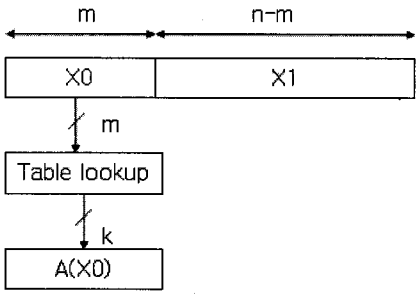

그림 4. 테이블 룩업 방식
Fig. 4 Table lookup method

그림 5는 2개의 표본점 (x_i, y_i)와 (x_{i+1}, y_{i+1})사이를 근사화하기 위해 2차 근사 다항식을 적용한 것을 나타낸다. 그림에서 $f(x)$는 정확한 함수 값을 의미하고, $P_2(x)$는 2차 근사 다항식을 나타낸다. $f(x)-P_2(x)$가 오차 값으로 이를 최소화하도록 적절한 계수 a, b, c를 결정하는 것이 중요하다.

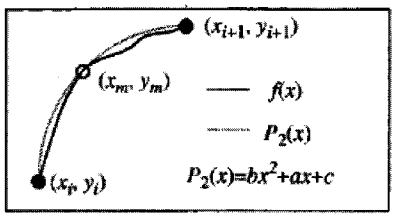

그림 5. 2차 다항식을 사용한 보간법
Fig. 5 Interpolation using second-order polynomial

최대 오차를 최소화하는 방식으로 Chebyshev 노드를 사용하는 근사 다항식 기법이 있지만, 이 방식은 구간 간격이 균등하지 않으므로 하드웨어 구현 측면에 있어서 비효율적이다. 본 연구에서는 하드웨어 구현이 용이한 Newton-Gregory Forward Difference 근사 다항식 방법을 사용하였다[9]. 입력 구간을 n개의 균등한 길이 h의 세부 구간으로 나눈 후, 그림 6과 같이 세부 구간의 1/2 지점 값 (x_m)의 정보(y_m)를 사용하는 방식으로 2차 근사 다항식은 식 (2)와 같다. 식 (2)는 [참고 문헌 8]에서 제시한 식을 하드웨어 구현이 용이하도록 변형한 식이다.

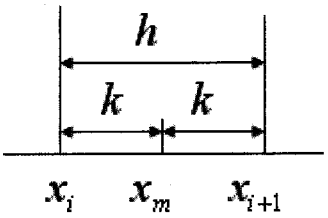

그림 6. 보간 다항식의 세부 구간
Fig. 6 Interval of interpolation polynomial

$$P(x)=\frac{s^2}{2}(y_{i+1}-2y_m+y_i)+\frac{s}{2}(y_{i+1}-y_i)+y_m \quad (2)$$
$$=\frac{s^2}{2}b+\frac{s}{2}a+x=\frac{s}{2}(a+sb)+c=s(\frac{a}{2}+\frac{b}{2}s)+c$$
$$=s(a^*+b^*s)+c^*$$

여기서 $b^*=(y_{i+1}-2y_m+y_i)/2=b/2,$
$a=(y_{i+1}-y_i)/2=a/2,$
$c^*=c=y_m=\frac{y_{i+1}+y_i-b}{2}=y^*_{i+1}+y^*_i-b^*$
$s=\frac{x-x_m}{k},$
$k=x_{i+1}-x_m=x_m-x_i=2^{-t}$

식 (2)에서 함수 값 y_i^*는 y_i의 1/2 값이다. y_i^*, y_{i+1}^*와 가장 복잡한 연산 계수인 b^*를 미리 계산해서 룩업 테이블(lookup table)에 저장해두고, 나머지 y_m값과 a^*, c^*계수는 연산 과정에 온라인 방식으로 계산하는 방식이다. 계수 b^*의 경우 $b^*=(y_{i+1}-2y_m+y_i)/2$ 가 0에 가까운 값이므로 상위 필드는 부호 확장 개념에서 동일 비트의 연속이므로, 부호 확장된 상위 필드를 저장하지 않는 방식으로 메모리 공간을 줄일 수 있다. 그리고 k 값이 2의 멱승 값이므로 k에 의한 나눗셈 동작은 간단한 이동 동작으로 구현할 수 있으므로, 복잡한 나눗셈이 필요없이 간단하게 구현 가능하다. $P(x)$는 이상적인 조건에서 식 (3)과 같이 2 단계로 이루어진다.

[단계 1] $temp=b^*\times s+a^*$ (3)
[단계 2] $P(x)=temp\times s+c$

3.3. 역코사인 프로세서의 하드웨어 설계

그림 3의 역코사인 함수의 2차 다항식 근사처리 단계에 사용되는 입력의 타당한 범위는 표 1과 같다. 외

부 입력 데이터는 부동 소수점 형식의 [-1,1]의 범위를 갖는데, 다항식 근사에 사용되는 값은 [0, 1) 사이의 고정 소수점 형식의 값을 갖는다. 선 처리 과정에 값의 부동 소수점 형식의 데이터를 고정 소수점 형식의 [0,1) 사이의 데이터로 조정하며, 입력 변수의 절대 값을 근사에 사용한다. 이렇게 입력의 절대 값을 근사에 사용할 경우 식(1)과 같이 음수인 경우 입력 데이터의 근사 연산 동작 후에 후 처리(post processing) 과정에서 보정을 하게 된다. 단, 입력 값이 +1.0, -1.0은 전처리 과정에서 등가 변환이 불가능하므로, 입력부에서 이러한 값을 특별한 조건으로 감지해, 후처리 과정에서 별도 처리된다.

표 1. 역코사인 함수의 근사를 위한 입력범위
Table 1. Input range for approximation of cos-1 function

$f(x)$	입력 범위	결과 값의 범위
$\cos^{-1}x$	[0, 1)	[0, $\pi/2$]

그림 7은 역코사인 프로세서의 전체 구조를 나타낸다.

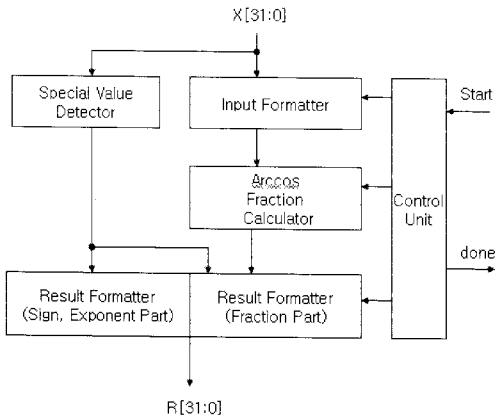

그림 7. 역코사인 프로세서의 블록도
Fig. 7 Block diagram of arccos processor

역코사인 프로세서는 입력 형식 변경 회로(Input Formatter), 특별 입력 감지기(Special Value Detector), 역코사인 근사 연산회로(arccos fraction calculator), 결과 생성 회로(Result Formatter), 제어 회로로 구성된다. 입력이 IEEE 부동 소수점 형식이므로, 역코사인 근사 연산 회로에 적합한 (0. 25) 형식의 고정 소수점 데이터로 변환이 필요하다. 여기서 (0.25) 형식이란 정수부가 0-비트, 소수부가 25-비트인 양의 고정 소수점 데이터 형식을 의미한다. 필요한 데이터 형식 변환을 수행하려면 부동 소수점 데이터의 소수부 값을 지수부의 반전(complement) 값으로 우측으로 이동시키면 된다. 단, 이 과정에 감춤 비트(hidden bit)가 입력으로 함께 사용된다. 결과 값으로 소수부 25 비트(0. 25 형식)만 결과로 취하게 된다. 그림 8은 입력 형식 변경 회로의 내부 구조를 나타낸다. 단, -1.0과 1.0은 근사계산에 필요한 타당한 입력 범위 [0, 1)을 벗어나므로, 특별 입력 감지기(Special Value Detector)에 의해 감지되었다가 최종 출력 변환 단계에서 처리된다.

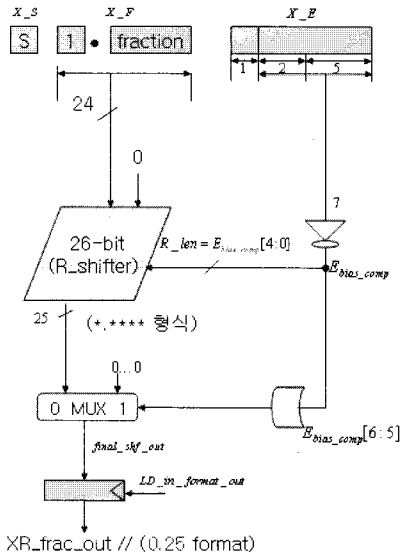

그림 8. 입력 형식 변경 회로
Fig. 8 Input Formatter

[0,1) 범위의 고정 소수점 데이터 형식에 대해 식 (2)에 따라 2차 다항식 보간 연산을 통해 역 코사인 값을 계산한다. 그림 9는 고정 소수점 형식의 입력 데이터에 대해 5 사이클에 역코사인 값을 계산하는 프로세서를 나타낸다. 식 (3)을 5개의 세부 단계로 나누어 구현하였다. 나누는 기준은 동작 주파수를 향상시키기 위해서 곱셈

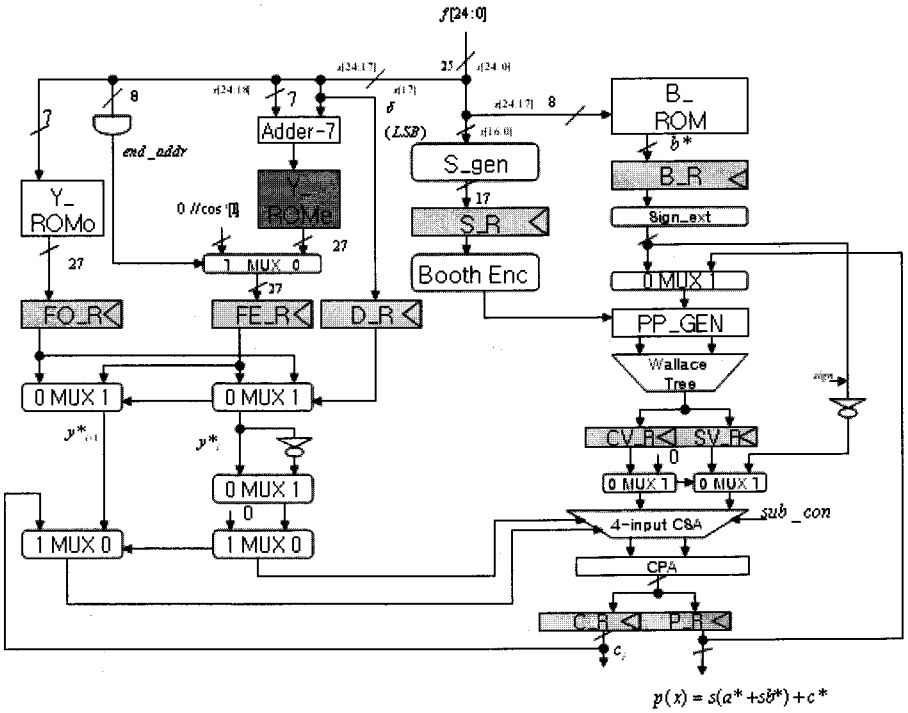

그림 9. 2차 근사 다항식을 이용한 역코사인 근사회로

Fig. 9 Cos^{-1} calculator based on 2nd order approximation polynomial

연산은 Wallace Tree를 통한 캐리 보존 벡터(carry vector, sum vector) 생성 단계와 덧셈 연산 단계로 나누었으며, ROM을 읽는 동작도 하나의 세부 단계로 할당하였다. 그리고 그림 9에서 Wallace Tree 단계와 덧셈 동작 단계가 레지스터로 분리되므로, MUX를 통한 입력 조정을 통해, 2개의 부분이 병렬로 수행 가능하다는 점을 활용하였다.

[단계 1] 계수 b^*, s와 y_i^*, y_{i+1}^* 값을 읽는 단계

$FE\_R \leftarrow Y\_ROM_e[i/2]$ //if $i = even$,
$FE\_R \leftarrow y_i^*$, else y_{i+1}^*
$FO\_R \leftarrow Y\_ROM_o[i/2]$ // if $i = even$,
$FO\_R \leftarrow y_{i+1}^*$, else y_i^*
$B\_R \leftarrow b\_ROM[i]$ // b^*
$s\_R \leftarrow \frac{x - x_m}{k}$ // $s \leftarrow s\_ \geq n$

[단계 2] $s \times b^*$의 곱셈 연산의 Wallace Tree 연산 단계와 c 계산 단계

$\{CV\_R, SV\_R\} \leftarrow s\_R \times B\_R$
// $[cv, sv] \leftarrow s \times b^*$
$C\_R \leftarrow FO\_R + FE\_R - B\_R$
// $c \leftarrow y_{i+1}^* + y_i^* - b^*$

[단계 3] temp 값 계산 단계

$P\_R \leftarrow CV\_R + SV\_R + FO\_R - FE\_R$
// $temp \leftarrow s \times b^* + a^*$

[단계 4] temp × s의 캐리 보존 곱셈 결과 생성 단계

$\{CV\_R, SV\_R\} \leftarrow s\_R \times P\_R$
// $cv, sv \leftarrow s \times temp$

[단계 5] 최종 결과 $P(x)$ 생성 단계

$P\_R \leftarrow CV\_R + SV\_R + C\_R$
// $P(x) \leftarrow temp \times s + c$

본 연구에서는 [0, 1]의 구간을 256개의 세부 영역으로 나누는 방식을 취했다. 그 결과 y_i^*값과 b_i^*값을 저장하는 공간으로 256개의 엔트리를 갖는 ROM이 필요하다. 그런데 y_{i+1}^*, y_i^*값이 동시에 사용되어야 하므로, 짝수 인덱스와 홀수 인덱스를 갖는 부분을 분리해서 2개의 ROM으로 구현하였다. 주소 값으로 입력 데이터의 상위 7-비트가 사용된다. 단, 인덱스 i는 입력 데이터의 상위 8 비트로 i가 짝수일 경우 y_i^* Y_ROMe에서 오며, y_{i+1}^*을 Y_ROMo에서 읽어오게 된다. 단, 인덱스가 홀수일 경우 y_i^*, y_{i+1}^*는 Y_ROMo과 Y_ROMe의 다른 행(주소)에 존재하게 되어. y_{i+1}^*는 주소를 (i/2)를 1만큼 증가시킨 후 Y_ROMe에서 읽게 된다. 그리고 256개의 세부 구간의 양 끝점의 함수 값(y^*)이 저장되어야 하므로, 실제로는 총 257개의 함수 값이 저장되어야 한다. 마지막 함수 값(y^{*256})을 Y_ROMe에 저장할 경우, ROM 구조가 바람직하지 않으므로, 마지막 경계치 값 ($y_{256}^* = \cos^{-1}(1) = 0$)을 ROM이 아닌 별도의 고정 상수로 배당하고, 인덱스(i)가 마지막 세부 구간을 가리킬 경우(그림 9에서 end_addr=1 조건), Y_ROMe값이 아닌 고정 상수 0을 선택하도록 하였다. 내부에 사용하는 ROM의 비트 수는 정밀도를 고려하여, Y_ROM의 경우 각 워드가 27-비트이고, B_ROM은 각 워드는 23-비트이다. B_ROM의 워드의 비트 수가 작은 이유는 모든 워드에 대해 공통적으로 상위 비트가 연속적인 1인 부분을 저장하지 않고, B_ROM을 읽는 과정에 부호 확장해서 사용하기 때문이다. 연산 과정에 발생할 수 있는 음수 데이터와 중간 연산 결과 범위를 분석하여 내부 레지스터와 하드웨어의 비트 폭을 고정 소수점 (2.25) 형식(부호 비트 1 비트, 정수부 1자리, 소수부 25-비트)을 유지하도록 하여, 연산 과정에 오버플로우에 따른 문제 발생을 배제하도록 하였다. 단, 최종 결과는 $\pi/2$ 보다 작은 양수 값이므로, 부호 비트를 제외한 (1.25) 고정 소수점 형식을 택해서, 최종 결과 형식 조정부로 보내진다. 결과 형식 조정부(Result Formatter)는 식(1)에 따라 입력 데이터의 부호 값에 따라 구분된 동작을 수행한다. 외부에서 제공되는 부동 소수점 입력 데이터의 부호가 양수인 경우, 단계 2에서 얻어진 고정 소수점 형태의 데이터를 즉각적으로 IEEE 표준 부동 소수점 형식에 맞는 데이터로 변환한다. 반면 입력 데이터가 음수 인 경우는 $\pi - \arccos(|x|)$에 따라 뺄셈 동작 후에 IEEE 부동 소수점 형식으로 변환 작업이 필요하다. 그런데 그림 2의 $\cos^{-1}x$ 함수 파형을 보면, 입력 데이터가 음수인 경우 결과 값의 범위가 $(\pi/2, \pi)$의 값을 가지므로, 정규화를 위한 최대 이동 거리가 0 혹은 1 비트 우측 이동 동작만 필요하다. 즉, 정규화 판단이 결과의 최상위 비트가 0인지 1인지를 통해 쉽게 결정이 가능하므로 정규화 동작과 뺄셈 동작을 단일 사이클에 구현할 수 있다. 이러한 정규화 과정을 통해 얻어진 값을 바탕으로 IEEE 부동 소수점 형식의 지수부 값을 쉽게 결정할 수 있다. 단, 부호 값은 항상 양수로 설정된다. 그림 10은 7개의 클록 사이클로 처리되는 설계된 역코사인 프로세서의 전체 동작을 나타낸다.

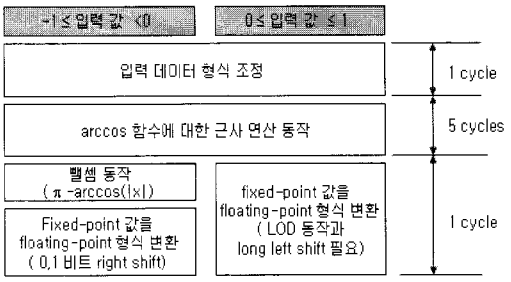

그림 10. 역코사인 함수의 연산 흐름도
Fig. 10 Operational flow of cos-1 function

입력 데이터의 부호 값이 음수인 경우 최종 결과 형식 조정부에의 경우 실수 값을 부동 소수점 데이터로 변환하는데, 0, 1비트 우측 이동만 필요하므로, 뺄셈 결과의 최상위 비트를 판단해 최종 지수부 값으로 bias+0 혹은 bias+1(bias 값=127)로 할당한다. 반면에 입력 데이터의 부호 값이 양수인 경우 다항식 근사에 의해 얻어진 결과에 대해 LOD(leading one detector)를 통해 왼쪽 이동 거리를 결정하여, 부동 소수점 형식에 맞는 정규화 동작을 수행하고, LOD에서 얻어진 이동 거리는 최종 지수부 값, bias - LOD값을 결정하는데 사용된다. 그리고 입력 데이터가 0, +1.0, -1.0인 경우는 특별 입력 감지부에 의해 감지되어, 각각 $\pi/2$, 0, π의 고정된 값으로 결정된다.

Ⅳ. 설계 검증 및 성능 분석

본 논문에서 역코사인 프로세서 설계를 위해 2가지 종류의 C 프로그램을 개발하였다. 하나는 전체 시스템을 검증하기 위한 C 프로그램으로 하드웨어 설계에 대한 테스트 벡터 생성과 하드웨어에 대한 검증 용도로 사용하였다. 단, 표준 C 라이브러리에는 역코사인 함수가 없으므로 $\tan^{-1}x$ 함수를 구현하는 atan 라이브러리를 활용하여 등가적으로 구현하였다. 다른 C 프로그램은 $\cos^{-1}x$ 함수의 2차 다항식 근사에 사용되는 3개의 계수 테이블, B_ROM, Y_ROMo, Y_ROMe에 내장되는 계수 값을 생성하는 프로그램이다. 회로 설계 작업 후에 역코사인 프로세서를 Verilog-HDL 언어로 작성 후 Modelsim 시뮬레이터를 사용하여 검증한 후, 그림 11과 같이 C 프로그램 모델에서 얻어진 결과와 비교하여 올바른 동작을 확인하였다.

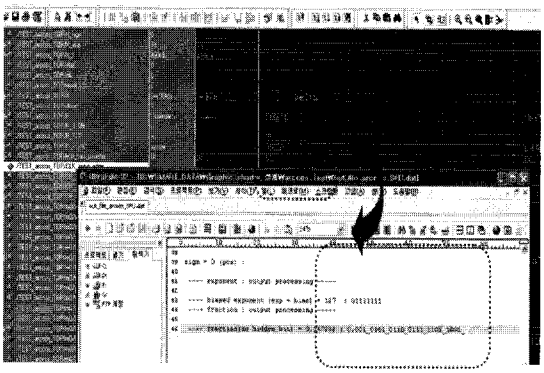

그림 11. 역코사인 프로세서에 대한 Modelsim 검증 파형 (입력 값 =0.392699)
Fig. 11 Modelsim waveform for arccos processor (input data =0.392699)

설계된 회로는 RTL 수준의 검증을 마친 후, 0.35μm 삼성 CMOS 표준 셀 라이브러리[10]과 SYNOPSYS 소프트웨어를 사용하여 합성한 결과 약 15,280개의 게이트로 구성되며, 타이밍 분석 결과 최악 전달 지연 시간은 7.58ns로, 최대 동작 주파수는 약 125 Mhz를 가짐을 알 수 있었다. 그림 12는 SYNOPSYS 소프트웨어로 합성한 역코사인 프로세서 회로를 나타낸다. 설계된 프로세서는 7 클록 사이클마다 1개의 결과를 출력하므로 연산 성능은 식 (4)와 같다.

$$execution\ rate = \frac{1}{7} \times 125 \approx 17.85\,MOPS \qquad (4)$$

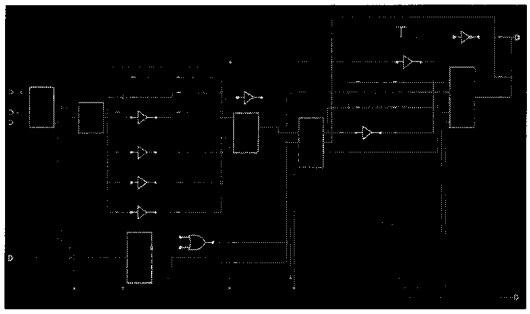

그림 12. 합성된 역코사인 프로세서
Fig. 12 Synthesized arccos processor

또한 C언어로 모델링한 결과와 설계된 회로의 출력 결과를 비교한 결과 2^{-22}까지 정확한 결과를 얻을 수 있었다. 설계한 역코사인 프로세서의 전기적 특성은 표 2와 같다. 설계한 역코사인 프로세서는 3개의 계수 테이블을 사용하며 2차 다항식 근사를 사용하여, 적절한 연산 성능과 높은 정밀도를 갖고 있어서 OpenVG를 비롯한 모바일 벡터 그래픽 응용에 효율적으로 적용 가능하다고 판단된다.

표 2. 전기적 특성
Table 2. Electric characteristics

공정	0.35 μm CMOS 공정
계산 알고리즘	룩업 테이블과 2차 다항식을 이용한 근사 방식
입출력 데이터	IEEE 단정도 부동 소수점 형식
게이트 수	15,280
latency	7 clock cycles
정밀도	2-22
동작 주파수	125 Mhz@3.3V
연산 성능	17.85 MOPS

V. 결 론

본 논문에서는 모바일 벡터 그래픽 분야에서 핵심 연산인 $\cos^{-1}x$ 함수를 처리하기 위한 연산 프로세서를 룩업 테이블 기반 2차 다항식 근사 기법을 사용하여 설계하고 검증하였다. 설계된 회로는 IEEE 표준 단정도 부동 소수점 형식의 데이터 표현을 가지며, 하드웨어를 공유하는 반복 연산 구조를 활용하며, 7 사이클에 연산을 수행한다. 그리고 모바일 그래픽 응용 분야에서 요구하는 정밀도를 고려하여 부동 소수점 데이터의 2 ulp(unit in the last position), 즉 2^{-22}의 정밀도를 갖는다. 설계된 회로는 Modelsim 소프트웨어로 검증하였으며 SYNOPSYS 소프트웨어로 합성한 결과 약 15,280개의 게이트 수로 구성되며, 약 125 Mhz의 동작 주파수로 동작 가능하다. 설계된 프로세서는 약 17.85 MOPS의 성능을 갖고 있으며, 적은 게이트 수와 높은 연산 정밀도를 갖고 있으므로 모바일 벡터 그래픽 분야에 적용이 가능할 것으로 판단된다. $\cos^{-1}x$ 함수에 구현에 사용한 연산 구조는 다른 초월 함수 구현에 적용 가능한 융통성이 있는 구조이므로. 2 또는 3 차원 그래픽 분야에서 필요한 다른 초월 함수에도 약간의 구조 변경을 통해 응용할 수 있다.

감사의 글

반도체 설계 교육센터(IDEC)의 CAD 소프트웨어 지원에 감사드립니다.

참고문헌

[1] Gaoqi He, Baogang Bai, Zhigeng Pan, and Xi Cheng, "Accelerated Rendering of Vector Graphics on Mobile Devices," Human-Computer Interaction, Part II, HCII 2007, LNCS 4551, pp.298-305, 2007.

[2] Sang-Yun Lee and Byung-Uk Choi, "Vector Graphic Reference Implementation for Embedded System," SEUS 2007, LNCS 4761, pp.243-252, 2007.

[3] Khronos Group Inc., OpenVG Specification Version 1.0.1, 2005.

[4] Volder, J.E., "The CORDIC Trigonometric Computing Technique," IEEE Trans. Elec. Comp., vol.EC-9, pp.227-231, 1960.

[5] Jean-Michel Muller, Elementary Functions-Algorithms and Implementation, 2nd edition, Birkhauser, 2006.

[6] Ping Tak Peter Tang, "Table-Driven Implementation of the Logarithm Function in IEEE Floating-Point Arithmetic," ACM Transactions on Mathematical Software, vol.4, no.16, pp.378-400, Dec. 1990.

[7] Michael J. Schulte and Earl E. Swartzlander, "Hardware Design for Exactly Rounded Elementary Functions," IEEE Transactions on Computers, vol.43, no.8, pp.964-973, August 1994.

[8] Jun Cao, and Belle W.Y. Wei, "High Performance Architectures for Elementary function generation," The 15th Symposium on Computer Arithmetic, pp.136-144, 2001.

[9] John H. Mathews, Numerical Methods for Computer Science, Engineering, and Mathematics, PH, chap.4, 1987.

[10] Samsung Electronics, STD90 /MDL90 0.35um 3.3V CMOS standard cell library for pure logic/ MDL Products, 2000.

저자소개

최병윤(Byeong-Yoon Choi)

1985년 2월 : 연세대학교 전자공학과 졸업
1992년 8월 : 연세대학교 전자공학과 공학 박사
2006년 1월 ~2006년12월: 오클랜드대학 방문 교수
1993년 3월~현재 : 동의대학교 교수
※관심분야 : RISC 마이크로프로세서 설계, 그래픽 및 암호 알고리즘의 SoC 설계

이종형(Jong-Hyoung Lee)

1987년 2월 : 연세대학교 전자공학과
졸업
2000년 8월 : 버지니아 공대
전기공학과 공학 박사

2003년 3월 ~ 현재 : 동의대학교 교수
※관심분야 : 광디바이스, SoC 설계