論文 86-23-3-8

RNS를 이용한 그래픽 데이터 스케일링

(Graphic Data Scaling with Residue Number Systems)

趙 源 敬*, 林 寅 七**

(Won Kyung Cho and In Chil Lim)

要 約

本 論文은 그래픽 디스플레이 데이타의 스케일링 演算을 위하여 RNS(Residue Number System)을 이용한 새로운 알고리즘을 提案한다.

이 알고리즘은 RNS에서 既存의 스케일링이 整數만에 의하여 可能하였던 것을 實數까지 擴長하였고, 베이스 擴長 알고리즘의 省略으로 高速의 스케일링 演算이 可能하도록 하였다.

提案된 알고리즘의 시뮬레이션에 의한 스케일링 計算 結果의 98%가 誤差範圍 1.3以內에 存在하였다. 또한 VLSI化를 考慮하여 同一한 構造의 演算 모듈의 配列로써 스케일링 프로세서의 構成이 可能하도록 하였다.

Abstract

This paper deseribes the design of a vector-coordinate rotation processor and the approximate evaluations of sine and consine based upon the use of residue number systems. The proposed algorithm results in a considerable improvement of computational speed as compared to the CORDIC algorithm. According to the results of computer simulation, the mean error of sine and cosine is 0.0025, and the mean error of coordinate rotation arithmatic is 0.65.

The proposed processor has the efficiency for the design and fabrication of integrated circuits, because it consists of an array of identical lookup tables.

I. 序 論

最近, 컴퓨터 그래픽스의 應用分野는 점점 擴大되어 가고 있으며, 특히 大規模 그래픽 데이타의 實時間處理 등에 있어서는 보다 高速 處理가 가능한 專用 프로세서를 필요로 하고 있다.

컴퓨터 그래픽에서 圖形은 점의 집합으로 표시되고, 2차원 直角 座標系에서 점의 위치는 두 정수(X, Y)에 의하여 결정된다. 실제 대부분의 應用分野에서 X, Y의 크기는 1024보다 작다. 컴퓨터 그래픽 디스플레이 데이타와 같이, 演算의 결과가 整數이고 整數의 범위가 크지 않은 경우에는 RNS를 이용하면 효과적인 演算 回路의 구성이 가능하다.

처음, RNS에 의한 연산 회로 구성에 관한 연구는 N. S. Szabo[6] 등에 의하여 일반적인 디지탈 演算 回路 구성을 대상으로 행해졌다. 그러나 RNS에서 整數의 加, 減, 乘算은 효율적이나, 除算과 實數 演算過程이 어렵기 때문에 현재는 디지탈 필터와 같은 특별한 분야에 제한적으로 이용되고 있다.[1][11]

本 論文에서는 RNS를 이용하여 그래픽 데이타의 高速 스케일링 프로세서를 設計하기 위한 알고리즘을 제안한다.

컴퓨터 그래픽에서 基本이 되는 演算은 圖形의 位置變換(translation), 스케일링(scaling), 回轉(ro-

*正會員, 慶熙大學校 電子工學科
(Dept. of Elec. Eng., Kyunghee Univ.)
**正會員, 漢陽大學校 電子工學科
(Dept. of Elec. Eng., Hanyang Univ.)
接受日字 : 1985年 9月 16日

tation)이다. 位置變換은 整數 演算만을 필요로 하기 때문에 RNS에서 용이하게 실현될 수 있다. 그러나 스케일링과 回轉 變換은 整數와 實數의 乘算을 필요로 하므로 RNS를 이용한 기존의 알고리즘[1-7]으로는 演算回路의 構成이 곤란하다.

本 論文에서는 새로운 實數 스케일링 알고리즘을 提案하고, 이 알고리즘을 이용함으로서 RNS를 사용한 高速 스케일링 프로세서가 용이하게 設計될 수 있게 한다.

Ⅱ. 理論的 背景

1. 2차원 그래픽의 演算

2차원 그래픽에서 자주 사용되는 基本 變換은 位置變換, 스케일링, 回轉 變換이다.

位置 變換의 基本式은

$$X' = X \pm Dx$$
$$Y' = Y \pm Dy \qquad (1)$$

로써 加, 減算만으로 용이하게 실현된다.

스케일링 變換의 基本式은

$$X' = X \cdot Sx$$
$$Y' = Y \cdot Sy \qquad (2)$$

로써 Sx, Sy가 實數일 경우 整數와 實數의 乘算이기 때문에 既存의 RNS 알고리즘으로는 어렵다.

回轉 變換의 基本式은

$$X' = X \cdot \cos\theta \mp Y \cdot \sin\theta$$
$$Y' = X \cdot \sin\theta \pm Y \cdot \cos\theta \qquad (3)$$

로써 (2)식과 같이 整數와 實數의 乘算과 加, 減算을 포함한다. (1), (2), (3)식에서 컴퓨터 그래픽의 가장 기본이 되는 演算은 整數의 加·減算과 整數와 實數의 乘算 결과중 整數 部分만 취하는(整數←整數×實數) 演算인 것을 알 수 있다.

2. RNS를 이용한 既存 스케일링 方式

$m=(m_1, m_2, \cdots, m_L)$을 서로 素(relatively prime integer)의 集合이라 하면

$$M = \prod_{i=1}^{L} m_i \qquad (4)$$

어떤 數, X가 $[0, M-1]$ 범위내의 整數이면

$$X = k_i \cdot m_i + x_i, \quad i = 1, 2, \cdots, L \qquad (5)$$

윗 식에서 x_i를 X의 i차 유수(residue)라 하고 $|X|_{m_i}$ 또는 $X \bmod m_i$라 표시한다. 그러면 X는 $(x_1, x_2, \cdots, x_L)$에 의하여 결정될 수 있다. 이것을 $X \cong (x_1, x_2, \cdots, x_L)$과 같이 표시한다. RNS의 特徵은 $X \cong (x_1, x_2, \cdots, x_L)$, $Y \cong (y_1, y_2, \cdots, y_L)$이고 $Z = X \circ Y = (z_1, z_2, \cdots, z_L)$이라 하면 z_i는 $z_i = |x_i \circ y_i|_{m_i}$로 된다(여기서 ○는 +, −, × 연산).

윗식의 결과로써 각 모듈러스(modulus)의 演算은 서로 獨立的이므로 캐리(carry) 情報가 필요없이 가능하기 때문에 모든 演算의 결과를 ROM이나 PAL에 기록(look up table)하는 방법에 의한 演算 回路의 구성이 가능하다.

RNS에서의 既存 스케일링 方法은 다음과 같다.

X를 入力이라 하고 Y를 스케일링 결과, K를 스케일링 因數라 하면

$$Y = \left\lfloor \frac{X}{K} \right\rfloor \qquad (5)$$

여기서 $[\bigcirc]$는 ○의 정수값을 의미한다. 윗 식을 다시 쓰면

$$X = Y \cdot K + |X|_K \text{ 이므로}$$
$$Y = \frac{X - |X|_K}{K} \qquad (6)$$

이다. K의 곱의 逆因數(multiplicative inverse), $\left|\frac{1}{K}\right|$를 다음과 같이 정의하면

$$|K \cdot |\frac{1}{K}|_{m_i}|_{m_i} = 1 \qquad (7)$$

Y의 i차 유수(residue) Yi는

$$Y_i = |Y|_{m_i} = \left| |X - |X|_{m_i}|_{m_i} \cdot \left|\frac{1}{K}\right|_{m_i} \right|_{m_i} \qquad (8)$$

와 같이 구할 수 있다.

윗 식에서 $\left|\frac{1}{K}\right|$가 존재하여야만 Y를 구할 수 있음을 알 수 있다.

Ⅲ. 實數 스케일링의 高速化를 위한 알고리즘의 提案

圖形의 縮小를 위해서는 實數 스케일링이 필요하다. RNS를 이용하여 實數 스케일링 演算이 가능하게 하기 위해서 다음과 같은 알고리즘을 제안한다.

實數 Y를 (9)식과 같이 표시하여 整數 Q, q, p에 의하여 그 값이 결정되도록 한다.

$$Y = Q + \frac{q}{p} \qquad (9)$$

(Y : 實數, Q, q, p : 整數, $q < p$)

실수 Y를 2진수로 표시하면

$$Y = \sum_{i=0}^{n} a_i \cdot 2^i + \sum_{i=1}^{m} a_{-i} \cdot 2^{-i} \qquad (10)$$

임의의 정수 X와 실수 Y의 곱은

$$X \cdot Y = X \cdot \left(\sum_{i=0}^{n} a_i \cdot 2^i + \sum_{i=1}^{m} a_{-i} \cdot 2^{-i} \right) \qquad (11)$$
$$= X \cdot Q + X \cdot q/p$$

(11)식에서 첫째 항, $X \cdot Q$는 整數와 整數의 곱이기 때문에 RNS에서 쉽게 구할 수 있다.

둘째항, $X \cdot q/p$의 계산에 RNS 스케일링 演算을 이용하기 위하여 q/p를 다음과 같이 변형한다.

$$q/p=\sum_{i=1}^{m} a_{-i}\cdot 2^{-i}$$
$$=q_1/m_L+q_2/m_L^2+\cdots+q_N/m_L^N \quad (12)$$

윗 식의 각 항은 RNS의 스케일링 알고리즘을 이용하여 계산할 수 있지만 스케일링 알고리즘에서 가장 문제가 되는 것은 MOD m_L의 레지듀를 구하기 위하여 많은 回數의 乘算과 減算을 필요로 하는 베이스 확장 알고리즘(base extension algorithm)이 필요한 것이다.

베이스 확장 연산을 생략함으로써 高速의 스케일링 연산을 할 수 있다.

베이스 확장 演算을 생략할 수 있게 하기 위하여 (12)식에서 $m_L=2^K$, $m=_{NK}$가 되도록 선택하면

$$q_1=2^K\cdot\sum_{i=1}^{k} a_{-i}\cdot 2^{-i}=\sum_{i=1}^{k} a_{-i}\cdot 2^{K-i}=\sum_{i=1}^{k} a_{-i}\cdot 2^{K-i}$$
$$q_2=2^{2K}\cdot\sum_{i=K+1}^{2K} a_{-i}\cdot 2^{-i}=\sum_{i=K+1}^{2K} a_{-i}\cdot 2^{2K-i}=\sum_{i=1}^{K} a_{-(K+i)}\cdot 2^{K-i}$$
$$\vdots \qquad\qquad \vdots \quad (13)$$
$$q_N=2^{NK}\cdot\sum_{i=(N-1)K+1}^{NK} a_{-i}\cdot 2^{-i}=\sum_{i=(N-1)K+1}^{NK} a_{-i}\cdot 2^{NK-i}$$
$$=\sum_{i=1}^{K} a_{-(N-1)K+i)}\cdot 2^{K-i}$$

으로 되어 q_i는 q/p를 2진수로 표시한 것을 K bits씩 분할한 값과 같다.

스케일링 프로세서를 파이프 라인 回路로 구성하기 위하여 (13)식을 이용하여 (11)식의 $X\cdot q/p$항을 다음과 같이 변형한다.

$$X\cdot q/p=X\cdot q_1\cdot 1/m_L+X\cdot 1/m_L\cdot q_2\cdot 1/m_L+\cdots$$
$$+X\cdot 1/m_L^{N-1}\cdot q_N\cdot 1/m_L$$
$$=\sum_{i=1}^{N} X\cdot 1/m_L^{i-1}\cdot q_i\cdot 1/m_L \quad (14)$$

(14)식에서 $X\cdot 1/m_L^{i-1}\cdot q_i$를 먼저 계산하고 $1/m_L$을 곱하는 것은 LSD(least significant digit)의 喪失에 의한 誤差를 줄이기 위한 것이다.

그림(1)은 (14)식을 계산하기 위한 回路 構成圖이다.

그림(1)에서 $X\cdot 1/m_L^{i-1}$을 계산하기 위한 MOD m_L의 Residue는 그림(2)에서와 같이 X를 K bits씩 분할한 값, Rx_i이므로 Rx_i를 계산하기 위한 베이스 확장 알고리즘의 실행이 생략된다.

$(X\cdot 1/m_L^{i-1}\cdot q_i)1/m_L$을 계산하기 위한 MOD m_L의 Residue는 그림(2)와 같이

$$|X\cdot 1/m_L^{i-1}\cdot q_i|_{m_L}=|Rx_i\cdot q_i|_{m_L}=Rq_i \quad (15)$$

로 되어 역시 베이스 확장 알고리즘이 필요없이 X의 2진수 표현을 Kbits씩 분할한 Rx_i와 q_i의 곱으로부터 쉽게 구할 수 있다.

Ⅳ. 回路 構成

回路 構成은 VLSI化에 용이하도록 各 演算 모듈의 獨立性, 演算 모듈의 規格化, 信號 흐름의 單純化를

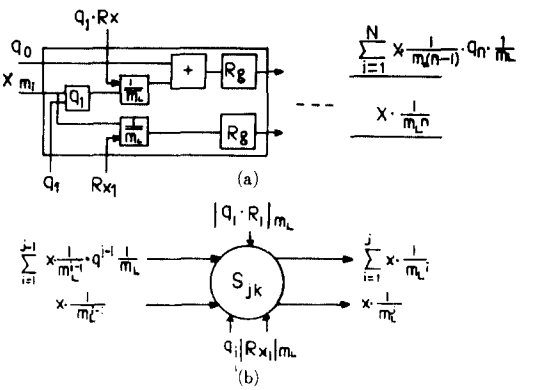


그림 1. 파이프 라인 구성을 위한 블록 다이아그램
(a) 연산 모듈 (b) 블록 다이아그램
Fig. 1. Block diagram for pipe-line processing.
(a) Arithmetic module. (b) Block diagram.

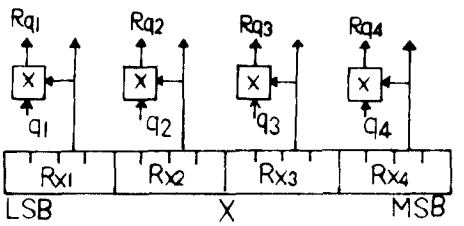


그림 2. K=4인 경우 Rx_i와 Rq_i
Fig. 2. Rx_i and Rq_i in the case of K=4.

고려하였다. (14)식은 ∑연산과, 연산식 중에 $1/m_L$, $1/m_L^2$, …, $1/m_L^{N-1}$을 순차적으로 필요로 하기 때문에 N단의 파이프 라인 방식의 回路 構成에 적당하다.

全體 回路 構成은 그림(3)과 같다. 그림(3)의 回路에서 매 클럭 펄스마다 1회씩 연속되는 그래픽 데이타의 스케일링 연산이 실행되어 高速 演算이 가능하다.

그림(3)의 回路는 並列 處理와 파이프라인 處理에 적합하고 同一한 構造의 演算 모듈은 ROM이나 PAL 등에 의하여 規則的이고 組織的인 設計가 가능하다.

Ⅴ. 誤差 解析

RNS의 스케일링은 연산 결과에서 정수만을 취하기 때문에 다음과 같은 誤差를 포함하게 된다.

(14)식의 각 항의 계산 과정은

$$X\cdot q_i\cdot\frac{1}{m_L}=\left(\frac{X\cdot q_i-R_i}{m_L}+\frac{R_i}{m_L}\right)$$
$$=\left\lfloor\frac{X\cdot q_i}{m_L}\right\rfloor_{m_L}+\frac{R_i}{m_L} \quad (16)$$

윗 식에서 $\lfloor\circ\rfloor$는 ○계산 결과의 整數 部分이고 $R_i<m_L$이다.

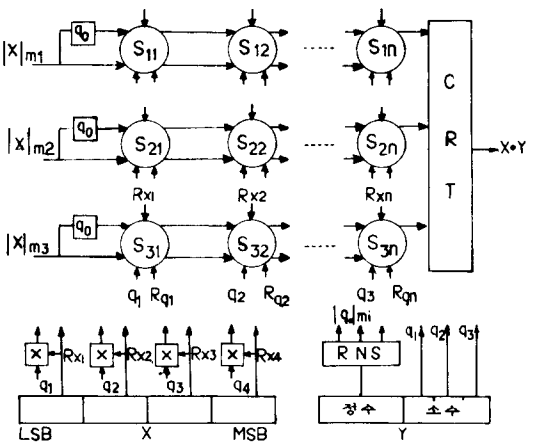


그림 3. 스케일링 프로세서의 구성
Fig. 3. Configulation of scaling processor.

$$X\cdot\frac{1}{m_L}\cdot q_2\cdot\frac{1}{m_L}=\left(\frac{X-Rx_1}{m_L}+\frac{Rx_1}{m_L}\right)\cdot q_2\cdot\frac{1}{m_L}$$

$$=\left(\left\lfloor\frac{X}{m_L}\right\rfloor\cdot q_2+\frac{Rx_1\cdot q_2}{m_L}\right)\cdot\frac{1}{m_L}$$

$$=\left\lfloor\frac{X\cdot q_2}{m_L^2}\right\rfloor+\frac{R_2}{m_L}+\frac{Rx_1\cdot q_2}{m_L^2}\qquad(17)$$

(17)식에서

$$\left\lfloor\frac{X\cdot q_2}{m_L^2}\right\rfloor=\left\lfloor\left\lfloor\frac{X}{m_L}\right\rfloor\cdot q_2\cdot\frac{1}{m_L}\right\rfloor$$

이다.

그림(3)의 스케일링 프로세서를 N단의 파이프 라인으로 구성한 경우, 윗 연산 과정을 일반화하면 i번째 항은

$$X\cdot 1/m_L^{i-1}\cdot q_i\cdot 1/m_L$$

$$=\lfloor X\cdot(1/m_L^i)\cdot q_i\rfloor+\frac{R_i}{m_L}+\sum_{j=1}^{i-1}Rx_j\cdot q_i/m_L^{i+1-j}\qquad(18)$$

RNS에서의 계산 결과는 [○]이기 때문에

$$[X\cdot q/p]=\left(\left\lfloor\frac{X\cdot q_1}{m_L}\right\rfloor+\left\lfloor\frac{X\cdot q_2}{m_L^2}\right\rfloor+\cdots+\left\lfloor\frac{X\cdot q_N}{m_L^N}\right\rfloor\right)\qquad(19)$$

그러면 誤差 ε는

$$\varepsilon=X\cdot q/p-[X\cdot q/p]\qquad(20)$$

이 된다.

(20)식에 (18)식의 결과를 대입하면

$$\varepsilon=\frac{R_1}{m_L}+\sum_{i=2}^{N}\left(\frac{R_i}{m_L}+\sum_{j=1}^{i-1}Rx_j\cdot q_i\cdot(1/m_L^{i+1-j})\right)\qquad(21)$$

윗 식에서 誤差는 演算 回路를 파이프 라인으로 구성하는 경우, 파이프 라인단수, N이 증가할 수록 커진다. (21)식에서 $R_i<m_L$이고 $\sum_{j=1}^{i-1}=Rx_j\cdot q_i\cdot(1/m_L^{i+1-j})<1$이기 때문에 最大 誤差 ε_{max}는 $\varepsilon_{max}<2N$이 된다.

誤差를 줄이기 위하여 (18)식을 다음과 같이 변형한다.

$$X\cdot 1/m_L^{i-1}\cdot q_i\cdot 1/m_L$$

$$=[X\cdot(1/m_L^i)\cdot q_i+\delta_i]-\delta_i+\frac{R_i}{m_L}+\sum_{j=1}^{i-1}Rx_j\cdot q_i/m_L^{i+1-j}\qquad(22)$$

여기서 $\delta_i=\begin{cases}0 & \text{if } R_i<R_T\\ 1 & \text{if } R_i\geq R_T\end{cases}$

(R_T는 0에서 m_L사이의 整數로써 誤差가 最小가 되도록 선택함).

그러면 (21)식은

$$\varepsilon_T=\frac{R_1}{m_L}-\delta_1+\sum_{i=2}^{N}\left(\frac{R_i}{m_L}-\delta_i+\sum_{j=1}^{i-1}Rx_j\cdot q_i\cdot(1/m_L^{i+1-j})\right)\qquad(23)$$

과 같이 된다.

(23)식의 오차 ε_T를 最小로 하는 R_T를 VI. 에서와 같이 시뮬레이션에 의하여 구하면 오차 ε_T를 최소로 할 수 있다.

VI. 시뮬레이션 결과 및 고찰

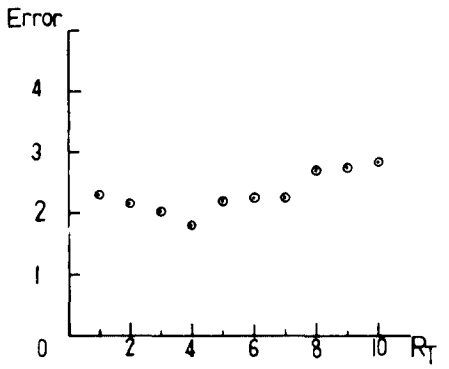
그림(3)의 스케일링 프로세서의 論理的인 妥當性과 誤差 解析을 위하여 제안된 알고리즘을 디지탈 컴퓨터를 사용하여 시뮬레이션하였다.

그림(3)의 각 演算 모듈은 서브루틴(subroutine) 으로 作成하였다. 실제 시뮬레이션에 適用한 데이타는 그래픽 디스플레이 데이타의 범위가 대부분 1024보다 작기 때문에 0～1023 범위의 데이타를 0.0에서 1.0 사이의 實數 스케일링 係數에 의하여 스케일링 하였다.

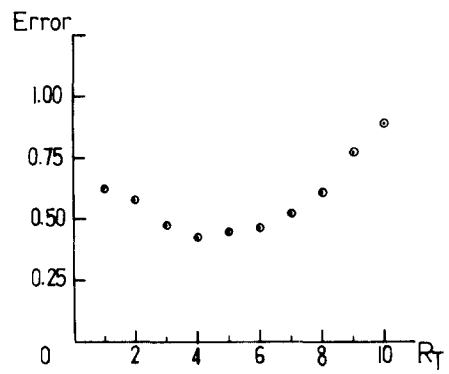
誤差를 最小로 하는 R_T를 구하기 위한 시뮬레이션 例는 $m_1=11$, $m_2=13$, $m_3=15$이고 $m_L=16$인 경우 實數 스케일링 係數를 0.0부터 1.0까지 0.1234씩 증가시키고 각각의 스케일링 係數에 0에서 1023까지 整數를 3씩 증가시키면서 計算 結果를 구하고 誤差를 最小로 하는 R_T를 구하였다.

誤差를 最小로 하는 R_T를 구하기 위하여 $m_L=16$인 경우, R_T에 따라 計算 結果의 98%를 포함하는 誤差의 範圍(그림(4－a)), R_T에 따른 平均誤差(그림(4－b))와 R_T에 따른 最大誤差(그림(4－c))를 그림(4)와 같이 그래프로 表示하였다.

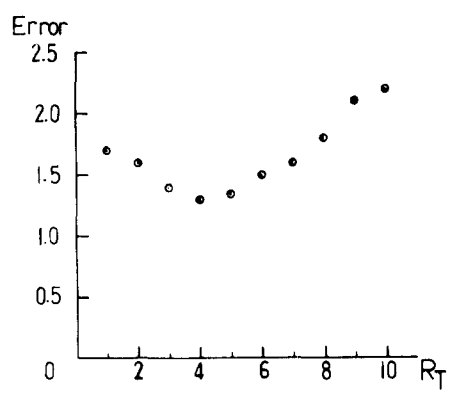
그 結果 $R_T=4$일때 計算 結果의 98%를 포함하는 誤差의 範圍가 1.30, 誤差의 平均은 0.434이고 最大 誤差는 1.83으로서 대부분의 스케일링 演算의 誤差가 1.30내에 存在하여 誤差가 最小가 되었다.



(a) R_T에 따른 계산 결과의 98%를 포함 하는 오차의 범위.
(a) The range of error including 98% of computing results with R_T.



(b) R_T에 따른 오차의 평균
(b) Mean of error with R_T.



(c) R_T에 따른 최대 오차
(c) Maximum error with R_T.

그림 4. R_T의 결정을 위한 시뮬레이션 결과
$m_1=11$, $m_2=13$, $m_3=15$, $m_L=16$, $N=3$
$i=0\sim1023$ $\triangle i=3$
$s=0.003\sim1.333$ $\triangle s=0.1234$
Fig. 4. Simulation results for decision of R_T.

Ⅶ. 結 論

그래픽 데이타의 高速 스케일링 演算을 위해 RNS를 사용한 새로운 스케일링 演算 回路를 제안하였다.

일반적으로 RNS에서는 제한된 數의 整數 스케일링만 가능한 것을 實數 스케일링이 가능하도록 하였다. 또한, 高速 演算을 위하여 RNS 스케일링에서 지금까지 필요로 하였던 베이스 확장 알고리즘을 생략할 수 있도록 하였다. 그 結果, 演算 回路를 파이프 라인 方式으로 構成하여 매 클럭마다 1 회씩 스케일링 演算이 可能하도록 하였다.

디지탈 컴퓨터를 사용하여 시뮬레이션 한 結果 $N=3$이고 $m_L=16$일때 計算 結果의 98%가 1.3 以下의 誤差 範圍내에 포함되었고 平均誤差는 0.434이었다.

또한, 提案된 回路는 各 演算 모듈간의 獨立性이 크고, 規格化된 演算 모듈을 사용하여 VLSI化가 용이하다.

提案된 回路에 制御 回路를 附加하여 일반적인 4칙 演算과 벡터 回轉 演算을 실행할 수 있도록 하여 컴퓨터 그래픽 分野뿐만 아니라 다른 디지탈 信號 處理 分野에 사용할 수 있는 高速 演算 回路를 構成하는 것은 필요한 課題라고 생각된다.

參 考 文 獻

[1] G.A. Jullien, "Residue Number Scaling and Other Operations Using ROM Array," *IEEE Trans. Comput.*, vol. c-27. no. 4. pp. 325-336, Apr. 1978.

[2] F.J. Taylor, "A VLSI Residue Arithmetic Multiplier," *IEEE Trans. Comput.*, vol. c-31, no. 6, Jun. 1982.

[3] K.H. O'keefe, "A Note on Fast Base Extention for Residue Number Systems with Three Moduli," *IEEE Trans. Comput.*, pp. 1132-1133, Nov. 1975.

[4] K.H. O'keefe and J.L.Wright, "Remarks on Base Extension for Modular Arithmetic," *IEEE Trans. Comput.*, vol. c-22, pp. 833-835, Sept. 1973.

[5] C.C. Guest, M.M. Mirsalethi and T.K. Gaylord, "Residue Number System Truth-table Look-Up Processing-Moduli Selection and Logical Minimization," *IEEE Trans. Comput.*, vol. c-33, no. 10, pp. 927-931, Oct. 1984.

[6] N.S. Szabo and R.I. Tanaka, *Residue Arithmetic and Its Applications to Computer Technology*, New York: McGraw-

Hill, 1967.

[7] E. Kinoshita, H. Kosako and Yojima, "Floating-point Arithmetic Algorithms in the Symmetric Residue Number System," *IEEE Trans. Comput.*, vol. c-23, pp. 9-20, Jan. 1974.

[8] J.D. Foley and A. Van Dam, *Fundamentals of Interactive Computer Graphics*, Addison-Wesley, 1982.

[9] C.A. Papachristou, "Direct Implementation of Discrete and Residue-Based Functions via Optimal Encoding: A Programmable Array Logic Approach," *IEEE Trans. Comput.*, vol. c-32, no. 10, pp. 961-968, Oct. 1983.

[10] M.A. Soderstrand and C. Vernia, "A High-Speed Low-Cost Modulo p_i Multiplier with RNS Arithmetic Applications," *Proc. IEEE*, vol. 68, pp. 527-532, Apr. 1980.

[11] W.K. Jenkins and B.J. Leon, "The Use of Residue Number Systems in the Design of Finite Impulse Response Digital Filter," *IEEE Trans. Circuits and Syst.*, vol. CAS-24, no. 4, pp. 191-201, Apr. 1977.

[12] Thu Van Vu, "Efficient Implimentations of the Chinese Remainder Theorem for Sign Detection and Residue Decoding," *IEEE Trans. Comput.*, vol. c-34, no. 7, Jul. 1985.