모바일 그래픽 가속기용 부동소수점 절사 승산기 설계

조용성* · 이용환**

Design of a Truncated Floating-Point Multiplier for Graphic Accelerator of Mobile Devices

Young-sung Cho* · Yong-hwan Lee**

본 논문은 정보통신부의 출연금으로 수행한 IT-SoC 핵심 설계 인력양성 사업의 수행결과입니다.

요 약

모바일 통신 서비스의 발전과 반도체 기술의 발달로 모바일 기기에 멀티미디어 서비스와 2D/3D 게임과 같이 고수준의 그래픽 처리를 필요로 하는 콘텐츠가 가능하게 되었다. 모바일 기기는 특성상 더욱 작은 칩 면적과 저전력 소비의 조건이 만족되어야 하며, 본 논문에서는 이러한 모바일 기기에 적용 가능한 2D/3D 벡터 그래픽 처리용 부동소수점 절사형 승산기를 설계한다. 본 논문의 승산기는 기본적으로 radix-4 Booth 인코딩을 적용하고, 면적과 전력 소모를 줄이기 위하여 절사방식을 사용한다. 구현된 절사형 승산기는 평균 퍼센트 오차가 0.00003% 정도로 모바일 기기에 충분히 적용가능하다. 승산기는 0.35um CMOS 셀 라이브러리를 이용하여 논리 합성되었고, 그 결과 절사되지 않은 기존의 radix-4 Booth 승산기에 비해 게이트 수가 약 33.8%정도 감소하였다.

ABSTRACT

As the mobile communication and the semiconductor technology is improved continuously, mobile contents such as the multimedia service and the 2D/3D graphics which require high level graphics are serviced recently. Mobile chips should consume small die area and low power. In this paper, we design a truncated floating-point multiplier that is useful for the 2D/3D vector graphics in mobile devices. The truncated multiplier is based on the radix-4 Booth's encoding algorithm and a truncation algorithm is used to achieve small area and low power. The average percent error of the multiplier is as small as 0.00003% and neglectable for mobile applications. The synthesis result using 0.35um CMOS cell library shows that the number of gates for the truncated multiplier is only 33.8 percent of the conventional radix-4 Booth's multiplier.

키워드

Floating-point, Multiplier, 2D/3D Graphics

Ⅰ. 서 론

기존의 모바일 환경에서의 그래픽은 단순한 이미지와 문자정보의 전달에 사용되었다. 그러나 모바일 통신 서비스의 발전과 함께 모바일 기기의 성능이 향상됨에 따라서 사용자들은 더 높은 수준의 GUI 환경, MMS의

* 엠텍비젼 연구원

** 금오공과대학교 전자공학부 조교수

접수일자 : 2007. 1. 24

지원, 2D/3D 게임과 같은 고수준의 그래픽 콘텐츠를 선호하게 되었다[1].

2D/3D 그래픽의 처리방식은 크게 래스터(raster) 기반의 이미지 처리 방식과 벡터(vector) 기반의 이미지 처리 방식으로 나뉜다. 래스터 그래픽스는 그래픽 장치에서 표현할 수 있는 최소 단위인 픽셀에 일대일로 대응되는 이미지 정보를 이용하여 그래픽을 표현한다. 벡터 그래픽스는 수학적인 표현 방법으로 정의된 이미지를 픽셀 데이터로 변환하여 그래픽을 표현한다. 따라서 벡터로 표현된 이미지 정보를 확대, 축소 및 회전 시에 이미지 손상이 없다. 이러한 장점으로 인해 벡터 그래픽스는 고수준의 그래픽 처리를 필요로 하는 멀티미디어 콘텐츠에 유리하다.

모바일 기기에서는 원활한 벡터 그래픽의 표현을 위해 별도의 가속장치를 필요로 하며[2] 모바일용 멀티미디어 콘텐츠의 활성화와 함께 모바일 그래픽 가속기 시장도 활성화 되고 있다. 본 논문에서는 모바일 벡터 그래픽 환경에 적합한 부동소수점 절사 승산기를 설계한다.

Ⅱ. 부동소수점 승산기 설계

단정도 부동소수점의 표현 방법에 따른 부동소수점 x와 y의 승산 결과 z는 다음 식 (1)과 같다.

$$Z = X \times Y = (-1)^{s_x \oplus s_y} \times (1.f_x \times 1.f_y) \times 2^{e_x + e_y} \tag{1}$$

부동소수점 승산 결과 값의 부호는 식 (2)와 같이 입력된 두 수의 부호에 XOR 연산을 취하여 쉽게 구할 수 있다. 결과 값의 지수와 가수는 다음 식 (3)과 (4)에서와 같이 각각 지수의 가산과 가수의 승산을 통해 구할 수 있다. 이때 식 (4)의 F_z는 재정규화 과정을 거치기 전의 결과를 의미한다.

$$S_z = S_x \oplus S_y \tag{2}$$

$$e_z = e_x + e_y \tag{3}$$

$$F_z = 1.f_x \times 1.f_y \tag{4}$$

단정도 부동소수점 지수 표현 $e = E - 127$에 따라 식 (3)을 다시 정리하면 식 (5)와 같다.

$$\begin{aligned} e_z &= e_x + e_y = E_x - 127 + E_y - 127 \\ &= E_x + E_y - 254 \end{aligned} \tag{5}$$

이때 출력되는 결과 값은 가수 부분의 예상 결과 값이 $1 \le F_z < 4$이므로 다음 식 (6)과 같다.

$$\begin{aligned} E_z &= E_x + E_y - 127, \text{ 단 } 1 \le F_z < 2 \\ E_z &= E_x + E_y - 126, \text{ 단 } 2 \le F_z < 4 \end{aligned} \tag{6}$$

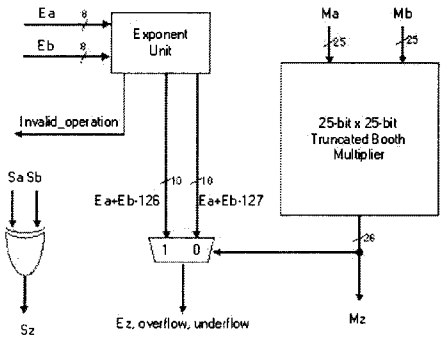

그림 1. 부동소수점 승산기의 블록도
Fig. 1 Block diagram of Floating-point multiplier

그림 1.은 앞서 유도된 식을 바탕으로 설계된 부동소수점 승산기 블록이다. 입력된 두 수의 부호에 XOR 연산을 취하여 결과 값의 부호를 얻는다. 지수 부분의 계산은 식 (6)과 같이 가수 부분의 결과 F_z에 따라 결과가 결정되므로 이때 발생하는 최대 지연 경로를 단축하고자 그림 1.에서와 같이 두 가지 경우에서 발생하는 값을 각각 출력하고 이를 F_z의 결과에 따라서 MUX를 통해 선택한다. 이때 overflow와 underflow도 같이 선택된다. 가수 부분의 연산은 절사 승산기를 이용하여 면적을 줄이는 방법을 사용한다.

Ⅲ. 절사 승산기를 이용한 가수의 연산

일반적으로, 두개의 N 비트의 승산의 결과는 2N 비트가 출력된다. 단정도 부동소수점의 경우 가수의 표현 범위는 정규화 비트를 포함하여 24비트로 한정된다. 따라

서 48비트의 승산 결과 중에서 상위 24비트만 사용하면 되므로, 하위 비트는 연산을 절사하는 절사승산기를 사용하면 승산기의 면적을 기존의 1/2로 줄일 수 있다[4]. 승산기의 입력이 n-bit일 경우 승수 X와 피승수 Y의 곱을 P라 하면 식 (7)과 같이 표현된다[3][5].

$$P = XY = \sum_{i=0}^{2n-1} p_i 2^i = \left(\sum_{i=0}^{n-1} x_i 2^i\right)\left(\sum_{i=0}^{n-1} y_i 2^i\right) \quad (7)$$

여기서 X와 Y의 곱인 P를 식 (8)과 같이 상위 부분(most-significant segment)인 MP와 하위 부분(least-significant segment)인 LP로 구분한다.

$$\begin{aligned} P &= MP + LP = X_h Y_h + X_h Y_l + X_l Y_h + X_l Y_l \quad (8) \\ &= \sum_{i=n}^{2n-1} P_i 2^i + \sum_{i=0}^{n-1} P_i 2^i \end{aligned}$$

그림 2.는 곱의 4가지의 영역 중에서 절사되는 부분을 진하게 표시하고 있다. 이러한 방법으로 하위 N-bit 영역을 생략하면 칩의 면적과 전력소모를 일반 승산기의 약 절반으로 줄일 수 있다. 그러나 하위 N-bit를 생략하는데 따른 승산결과에 절사오차가 발생하므로 이를 보상해 주기 위한 별도의 방법이 필요하다.

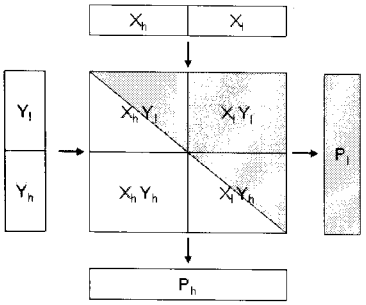

그림 2. 곱셈 결과에서 절사되는 부분
Fig. 2 Truncated area of product

절사형 승산기의 오차보상 방법은 Kidambi가 제안한 확률적 접근 방법에 기초한 고정 오차 보상 방법[5]과 Jou가 제안한 근사화된 캐리의 합을 이용하여 오차 보상 회로를 만드는 가변 오차 보상 방법[6]이 있다. 본 논문에서는 부동소수점 승산기를 radix-4 Booth 인코딩을 통하여 설계하고 절사된 부분의 오차 보상은 가변 오차 보상법을 이용하여 구현한다.

Radix-4 Booth 인코딩은 승수를 3-bit씩 묶어 그 비트 패턴에 따라 부분곱을 생성함으로써 부분곱의 수를 절반으로 줄이는 방법이다. 일반적으로 n-bit 승수 Y에 대한 radix-4 Booth 인코딩은 식 (9)와 같이 표현된다[7-8].

$$\begin{aligned} Y &= \sum_{k=0}^{\frac{n}{2}-1} (-2y_{2k+1} + y_{2k} + y_{2k-1}) \times 2^{2k} \\ &= \sum_{k=0}^{\frac{n}{2}-1} Q(k) \times 2^{2k}, \; y_{-1} = 0, \; Q(k) \in \{-2, -1, 0, 1, 2\} \end{aligned} \quad (9)$$

다음 표 1.과 그림 3.은 본 논문에서 사용한 radix-4 Booth 인코딩 표와 디코더를 나타낸다. 이때 표 1.의 add 부분은 절사되는 영역에 속하므로 별도의 디코더를 만들지 않았다.

표 1. 본 논문에서 사용한 radix-4 Booth 인코딩
Table 1. Radix-4 Booth's encoding used in this paper

b_{2n+1}	b_{2n}	b_{2i-1}	Q(k)	zero	y2	neg	add
0	0	0	0	0	1	0	1
0	0	1	1	1	0	0	0
0	1	0	1	1	0	0	0
0	1	1	2	1	1	0	0
1	0	0	-2	1	1	1	1
1	0	1	-1	1	0	1	1
1	1	0	-1	1	0	1	1
1	1	1	0	0	1	1	1

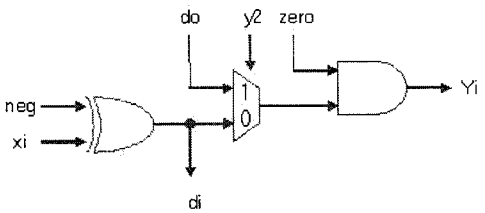

그림 3. 본 논문에서 사용한 Booth 디코더
Fig. 3. Booth's decoder used in this paper

그림 4.는 8-bit × 8-bit의 Booth 절사 승산기를 나타내고 있으며 진하게 표시된 하위 8-bit를 절사하게 된다. 각 부분곱은 확장부호 제거기법을 이용하여 부호 비트의

불필요한 연산을 줄인다. 실제 부분곱을 결정하는 $Q(k)$는 식 (9)와 같이 $-2 \le Q(x) \le 2$의 범위를 갖는다. w_k는 $Q(k)=2$이고 $x_{n-1}=1$ 이거나 $Q(k)=-2$이고 $x_{n-1}=0$인 경우를 의미하며, z_k는 $Q(k)=1$이고 $x_n=1$ 이거나 $Q(k)=-1$이고 $x_n=0$인 경우를 의미한다.

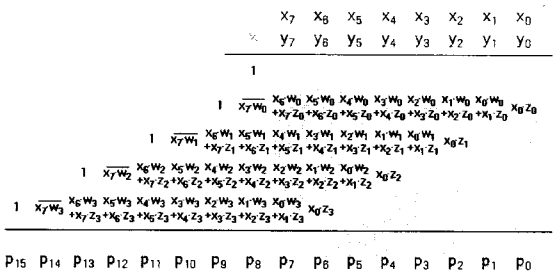

그림 4. 8-bit × 8-bit Booth 승산기
Fig. 4 8-bit × 8-bit Booth's multiplier

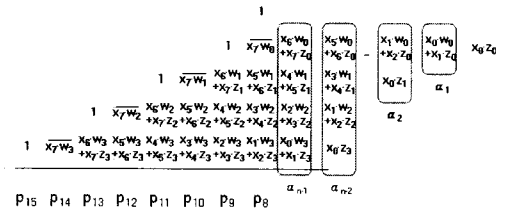

그림 5. 절사된 Booth 승산기의 캐리 발생 알고리즘
Fig. 5 Carry generation algorithm of the truncated multiplier

Booth 승산기에서 발생하는 부분곱의 세로 열로부터 발생되는 캐리의 합을 α_i라고 정의한다면 α_i는 그림 5와 같이 표현된다. 이를 수식으로 표현하면 식 (10)과 같다.

$$\alpha_{n-1} = \left\lfloor \frac{1}{2}(x_{n-1} \cdot z_0 + x_{n-2} \cdot w_0 + \cdots \cdots + x_1 \cdot z_k + x_0 \cdot w_k + \alpha_{n-2}) \right\rfloor \quad (10)$$

α_{n-2} 또한 식 (11)과 같이 정의할 수 있다.

$$\alpha_{n-2} = \left\lfloor \frac{1}{2}(x_{n-2} \cdot z_0 + x_{n-3} \cdot w_0 + \cdots \cdots + x_0 \cdot z_k + \alpha_{n-3}) \right\rfloor \quad (11)$$

이같은 전개 과정을 통하여 α_{n-1}을 확장하면 식 (12)로 표현된다.

$$\alpha_{n-1} = \left\lfloor \begin{array}{l} \frac{1}{2}(x_{n-1} \cdot z_0 + x_{n-2} \cdot w_0 + \cdots \\ \quad \cdots + x_1 \cdot z_k + x_0 \cdot w_k) \\ + \frac{1}{4}(x_{n-2} \cdot z_0 + x_{n-3} \cdot w_0 + \cdots + x_0 \cdot z_k) \\ + \cdots + \frac{1}{2^{n-1}}(x_1 \cdot z_0 + x_0 \cdot w_0) \end{array} \right\rfloor \quad (12)$$

식 (12)를 간단히 하기 위해서 β를 아래와 같이 정의한다.

$$\beta = \sum_{k=0}^{\frac{N}{2}} (x_{n-1-2k} \cdot z_k + x_{n-2-2k} \cdot w_k)$$
$$= x_{n-1} \cdot z_0 + x_{n-2} \cdot w_0 + \cdots + x_1 \cdot z_k + x_0 \cdot w_k$$
단 x_n에서 $n \ge 0$ (13)

β를 이용하여 식 (12)를 간단히 하면 식 (14)와 같다.

$$\alpha_{n-1} = \left\lfloor \frac{1}{2}\beta + \lambda \right\rfloor \quad (14)$$

여기서 λ는 식 (15)와 같이 정의된다.

$$\lambda = \frac{1}{4}x_{n-2} \cdot z_0 + \frac{1}{4}x_{n-3} \cdot w_0 + \cdots \cdots + \frac{1}{2^{n-1}}x_1 \cdot z_k + \frac{1}{2^{n-1}}x_0 \cdot w_k \quad (15)$$

식 (14)와 같이 절사된 부분의 합 α_{n-1}를 정의할 경우 β와 λ와의 관계를 밝히면 원하는 α_{n-1}을 구할 수 있게 된다. β는 다음과 같은 순서를 통해 그 값을 유추할 수 있다.

가) 주어진 β에서 w_k와 z_k는 Radix-4 Booth 인코딩 결과 값이므로 동시에 1이 될 수 없고 확률이 0 이므로 β는 $\beta = \beta_z + \beta_w$로 표현할 수 있다. β의 연산에 입력되는 피 승수인 x_{n-1-2k}와 x_{n-2-2k}가 1이 될 확률이 0.5이며, Radix-4 Booth 인코딩 결과 값인 w_k가 1이 될 확률은 0.25, z_k가 1이 될 확률은 0.5이다.

나) β_z가 1이 되는 경우의 수는 $2\beta \le N(\beta_z) \le n+\beta$와 같으

며, β_w가 1이 되는 경우의 수는 $3\beta \le N(\beta_w) \le 2n+\beta$와 같다.

다) 따라서, β_z와 β_w에서 하나의 입력 비트가 1이 될 확률은 각각 $(n+3\beta)/4n$과 $(n+2\beta)/3n$으로 표현할 수 있다.

라) β_z가 1이 될 경우의 확률을 $p1_z(x_i z_j)$라 하고 β_w가 1이 될 경우의 확률을 $p1_w(x_i w_j)$라 하면 그 확률은 식 (16)과 같이 정의할 수 있다.

$$p1_z(x_i z_j) = \frac{(n+3\beta)^2}{16n^2},$$

$$p1_w(x_i w_j) = \frac{(n+2\beta)^3}{27n^3} \tag{16}$$

마) 그림 6.의 결과를 분석해 보면 $p1_z(x_i z_j)$, $p1_w(x_i w_j)$이 발생할 확률을 식 (17)과 같이 근사화 할 수 있다.

$$p1_z(x_i z_j) \cong p1_w(x_i w_j) \cong \frac{\beta}{n} \tag{17}$$

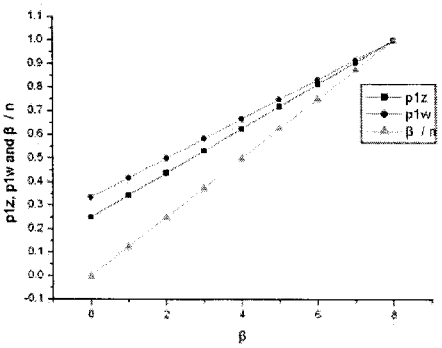

그림 6. n=8인 경우 $p1_z$, $p1_w$, β/n의 확률
Fig. 6 Probabilities of $p1_z$, $p1_w$, and β/n, where n=8

바) 마)에서 계산된 확률을 기준으로 λ는 식 (18)과 같이 구할 수 있다.

$$\begin{aligned}\lambda &= \frac{1}{4}x_{n-2} \cdot z_0 + \frac{1}{4}x_{n-3} \cdot w_0 + \cdots \\ &\quad \cdots + \frac{1}{2^{n-1}}x_1 \cdot z_0 + \frac{1}{2^{n-1}}x_0 \cdot w_0 \\ &= \frac{1}{4}p1_z(x_{n-2}z_0) + \frac{1}{4}p1_w(x_{n-3}w_0) + \cdots \\ &\quad \cdots \frac{1}{2^{n-1}}p1_z(x_1 z_0) + \frac{1}{2^{n-1}}p1_w(x_0 w_0) \\ &= p1z(x_i, z_j) \times \left(\frac{n-1}{4} + \frac{n-2}{8} + \cdots + \frac{2}{2^{n-1}}\right) \\ &= \frac{\beta}{n} \times \left(\frac{n}{2} - 1\right) = \frac{\beta}{2} - \frac{\beta}{n}\end{aligned} \tag{18}$$

사) 바)의 과정을 통해 계산된 β와 λ를 바탕으로 α_{n-1}를 식 (19)와 같이 근사화 할 수 있다.

$$\alpha_{n-1} \cong \acute{\alpha} \equiv \begin{cases} 0, & \text{if } \beta = 0 \\ \beta - 1, & \text{if } \beta > 0 \end{cases} \tag{19}$$

그림 7.은 본 논문에서 부동소수점 가수 부분의 승산에 사용된 25비트 Booth 절사 승산기를 나타낸다. 단정도 부동소수점의 가수는 23비트로 이루어져 있으며 최상위 비트에 '1'의 값이 정규화 되어있어 연산시에는 24비트로 확장이 필요하다. 또한 부호 확장 제거 기법을 사용하기 위하여 25비트로 부호를 확장하여 계산하였다. 그림 7.의 박스로 처리된 부분이 절사된 부분에서 발생하는 캐리를 보상하는 회로이다. 이때 carry_in 신호는 β 값이 '0'일 경우 '1'로, '1'일 경우 '0'으로 결정된다.

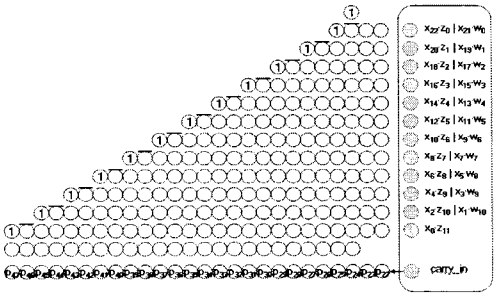

그림 7. 25-bit × 25-bit 가수 승산
Fig. 7 Multiplication of 25-bit × 25-bit fractions

Ⅳ. 설계 검증 및 성능 평가

4.1 기능 검증 및 오차 분석

본 논문에서 설계된 부동소수점 승산기는 기존의 부동소수점 연산기에 비해 적은 면적으로 구현되지만 매우 정확한 연산 결과는 아니다. 따라서 부동소수점 승산기의 기능 검증과 연산 결과에 대한 오차 분석이 필요하다.

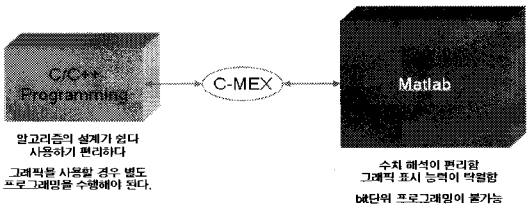

그림 8. 알고리즘의 검증 방법
Fig. 8 Algorithm verification method

그림 8.은 본 논문을 위한 알고리즘 수립단계에서 사용한 방법을 나타내고 있다. C 언어를 이용하여 하드웨어 구현을 전제로 한 부동소수점 승산기 알고리즘을 설계하고, 이를 수치 해석이 편리한 Matlab을 이용하여 알고리즘의 성능 평가를 우선 수행하였다. C 언어와 Matlab 사이의 인터페이스는 Matlab에서 지원하는 인터페이스 라이브러리인 C-MEX를 이용한다.

소프트웨어 가속기를 통하여 검증된 알고리즘은 Verilog로 하드웨어 모델링을 수행하였으며 설계된 부동소수점 승산기는 그림 9.와 같은 방법을 통해 기능 검증 및 오차 분석을 수행하였다.

그림 9. 기능 검증 및 오차 분석 흐름도
Fig. 9 Functional verification and error analysis flow

기능 검증 및 오차 분석에 사용되는 입력 벡터는 그림 9.와 같이 C 프로그램을 통해 랜덤하게 생성되며, 이 때 생성된 입력 벡터를 testbench를 통하여 실제 연산 결과와 비교하여 기능 검증을 수행하였다. 오차 분석 프로그램은 testbench를 통해 생성된 결과 벡터를 이론값과 비교하여 실제 부동소수점 승산기의 오차 특성을 분석하였다.

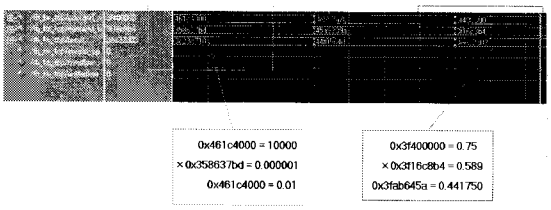

그림 10. 부동소수점 승산기의 기능 검증
Fig. 10 Functional verification of Floating-point multiplier

본 논문에서는 설계된 부동소수점 승산기의 기능 검증과 오차 분석을 위하여 C 프로그램을 이용하여 생성된 50,000개의 테스트 벡터를 그림 10.과 같이 모델심을 이용하여 기능 검증을 수행하였으며 연산 결과에서 발생하는 오차는 식 (20), (21)과 같이 평균 오차와 평균 퍼센트 오차를 이용하여 연산 정밀도를 분석하였다.

$$Err_{avr} = \frac{\sum |PM - PV|}{M} \tag{20}$$

$$PErr_{avr} = \frac{\sum \left(\frac{|PM - PV|}{PM} \right)}{M} \times 100 \tag{21}$$

식 (20)과 (21)에서 PM은 이론적인 연산 결과를 의미하며 PV는 본 논문에서 제안한 알고리즘을 이용하여 설계된 하드웨어를 통해 출력된 결과를 의미한다. M은 입력된 테스트 벡터의 개수를 나타낸다.

위와 같은 오차분석 방법을 통한 부동소수점 승산기의 절사 영역에서 발생된 평균 오차는 0.000001이고 평균 퍼센트 오차는 0.00003%로 매우 적어 벡터의 연산의 정확도에 크게 영향을 주지 않는다.

4.2 부동소수점 승산기의 성능평가

본 논문에서는 부동소수점 승산기를 0.35㎛ CMOS 셀 라이브러리와 Design Compiler를 이용하여 논리 합성을 수행하였다.

표 2. 절사승사기와 기존 radix-4 Booth 승산기의 면적 비교(25-bit X 25-bit)
Table 2. Area comparison between the truncated multiplier and the conventional radix-4 Booth's multiplier (25-bit X 25-bit)

공정		0.35㎛ Standard Cell
합성조건	Voltage	2.7V(typical: 3.3V)
	Process	1.3(typical: 1)
	Temperature	125.0(typical: 25)
	Interconnect Model	worst case tree
동작 주파수		61MHz@2.7V
면적	Radix-4 Booth 승산기	5407.0
	절사 승산기	3577.2
면적감소 효과		33.8%

일반적인 부동소수점 승산기는 가수의 승산에 25-bit의 radix-4 Booth 승산기를 사용한다. 이때 표 2.의 조건으로

합성한 결과 가수 부분의 승산기의 면적은 5407.0 게이트로 합성되었고, 가수 승산에 불필요한 하위 비트를 절사함으로써 약 33.8%의 면적 감소 효과를 확인할 수 있었다.

V. 결 론

본 논문에서는 모바일용 그래픽 가속기를 효과적으로 지원하기 위한 부동소수점 절사 승산기를 설계하였다. 저면적 특성을 갖는 부동소수점 승산기의 설계를 위하여 radix-4 Booth 알고리듬과 가수 부분의 절사 승산기를 사용한 모델을 구현하였고 절사 승산기에 의해 절사된 영역의 오차 보상을 위하여 가변오차 보상법을 적용하였다.

위와 같이 정의된 알고리즘을 설계된 소프트웨어 그래픽 가속기를 통해 알고리즘을 검증하였고, Verilog로 모델링을 수행하여 모델심과 오차 보상 프로그램을 이용하여 기능 검증과 오차 분석을 수행하였다. 이러한 과정을 거쳐 설계된 부동소수점 절사 승산기는 0.35㎛의 CMOS 셀 라이브러리를 통해 논리 합성을 수행한 결과 기존 radix-4 Booth 승산기를 사용한 결과와 비교하여 33.8%의 면적 감소효과를 얻을 수 있었다.

본 논문에서 설계된 부동소수점 절사 승산기는 높은 데이터 처리능력에도 불구하고 상당한 저면적 특성을 가져 모바일 기기의 그래픽 연산에 널리 활용될 수 있을 것으로 판단된다.

※ 반도체설계교육센터(IDEC)의 CAD Tool 지원에 감사드립니다.

참고문헌

[1] 길상철, 김석진, 나도백, 박태근, "모바일 3차원 그래픽 가속기 SoC 기술 동향", 『KISTI 기술정보분석보고서』, 66 면, Dec. 2004

[2] Troy Evans, "Introducing Macromedia Flash Lite 1.1", Macromedia

[3] I. Koren, "Computer Arithmetic Algorithm Second Edition", A K Peters, Ltd., Nov, 2001

[4] 정해현, 박종화, 신경욱, "저전력 디지털 신호처리 응용을 위한 작은 오차를 갖는 절사형 Booth 승산기 설계", 한국해양정보통신학회논문지, vol. 6, no. 2, pp. 323-329

[5] S.S. Kidambi, F. Guibaly and A. Antoniou, "Area-efficient multipliers for digital signal processing applications", IEEE Trans. on CAS-II, vol. 43, no. 2, pp. 90-95, Feb. 1996

[6] J.M. Jou, S.R. Kuang, and R.D. Chen, "Design of low-error fixed-width multipliers for DSP application", IEEE Trans. on CAS-II, vol. 46, no. 6, pp. 836-842, Jun. 1999

[7] A.D. Booth, "A signed binary multiplication technique", Quarterly J Mechanics, Appl. Math, vol. 4, part 2, pp. 236-240, 1951

[8] L.P. Rubinfield, "A proof of the modified Booth's algorithm for multiplication", IEEE Trans. on Computers, vol. C-24, no. 10, pp. 1014-1015, Oct. 1975

저자소개

조 용 성(Young-sung Cho)

1998.3～2003.2 금오공과대학교 전자공학과 졸업
2005.3～2007.2 금오공과대학교 전자공학과 공학석사
2007년 3월～ 현재 엠텍비젼 연구원
※관심분야 : SoC 설계, 반도체 IP 설계, Camera Signal Processor(CSP)

이 용 환(Yong-hwan Lee)

1993. 2. 연세대학교 전자공학과(공학사)
1999. 2. 연세대학교 전자공학과(Ph.D.)
1999～2002 하이닉스반도체
2003～2004 삼성전자
2004～ 금오공과대학교 전자공학부
※관심분야 : SoC, 임베디드 시스템 및 소프트웨어, 마이크로프로세서 구조