논문 2009-46SD-3-3

모바일 3차원 그래픽 연산을 위한 제곱근 및 역제곱근 연산기 구조 및 설계

(Design of Square Root and Inverse Square Root Arithmetic Units for Mobile 3D Graphic Processing)

이 찬 호*

(Chanho Lee)

요 약

본 논문에서는 모바일 환경 기반의 3차원 그래픽 연산을 위한 조명처리 엔진 및 쉐이더 프로세서에 사용 가능한 제곱근과 역제곱근 연산기의 구조를 제안한다. 제안하는 구조는 Taylor 전개식을 기반으로 하여 참조 테이블 및 보정 유닛으로 구성되어 있어 참조 테이블의 크기를 줄였다. 연산 결과는 IEEE-754 표준의 단정도 32 bit 부동소수점 형식과 모바일 환경을 위하여 이를 축소한 24 bit 부동소수점 형식에 대해 OpenGL 1.x ES 에서 요구하는 10^{-5}의 정확도를 거의 만족한다. 제안된 구조에 따라 설계된 제곱근 및 역제곱근 연산기는 Verilog-HDL을 사용하여 설계되었으며 파라미터 변경을 통하여 24 bit와 32 bit 연산이 가능하도록 합성이 가능하고 1사이클의 잠복기를 갖는다. 설계된 연산기들의 동작은 FPGA를 이용한 검증시스템을 통하여 검증하였다.

Abstract

We propose hardware architecture of floating-point square root and inverse square root arithmetic units using lookup tables. They are used for lighting engines and shader processor for 3D graphic processing. The architecture is based on Taylor series expansion and consists of lookup tables and correction units so that the size of look-up tables are reduced. It can be applied to 32 bit floating point formats of IEEE-754 and reduced 24 bit floating point formats. The square root and inverse square root arithmetic units for 32 bit and 24 bit floating format number are designed as the proposed architecture. They can operation in a single cycle, and satisfy the precision of 10^{-5} required by OpenGL 1.x ES. They are designed using Verilog-HDL and the RTL codes are verified using an FPGA.

Keywords : 3D, lighting, inverse square root, square root, implementation

I. 서 론

최근 3차원 그래픽 기술은 시각적 눈높이가 높아진 사용자들을 만족시키기 위하여 게임, 네비게이션 시스템, GUI(Graphic Use Interface) 등의 다양한 응용분야에서 보다 화려하고 사실적인 효과를 표현하기 위한 방법으로 주목받고 있다. 3차원 그래픽 기술은 수 많은 부동소수점 연산을 필요로 하기 때문에 일반적으로 보다 빠른 연산을 위한 하드웨어 가속기를 필요로 하며 이러한 3차원 그래픽 하드웨어 가속기는 그 구조 및 처리 기술, 그리고 내부 연산기의 연산 속도 등의 다양한 요소들에 의해 성능이 결정된다.

모바일 기반의 3차원 그래픽 기술은 전력과 자원이 제한적이라는 점에서 데스크탑 환경의 3차원 그래픽 기술과 차별화 되며, 이는 OpenGL 이라는 3차원 그래픽 API(Application Programming Interface) 표준을 제공하는 Khronos 그룹에서 모바일 환경을 위하여 OpenGL

* 정회원, 숭실대학교 정보통신전자공학부
(School of Electronic Engr., Soongsil University)
※ 본 논문은 숭실대학교 교내연구비 지원을 받았습니다.
접수일자: 2007년8월28일, 수정완료일: 2009년2월13일

ES 라는 축소된 표준을 제시하는 것에서도 확인할 수 있다[1]. 따라서 모바일 기반의 3차원 그래픽 하드웨어 가속기를 구성하는 연산기들은 모바일 환경의 특성상 그 구조와 동작이 제한적일 수밖에 없다. 이러한 모바일 환경에 적합한 부동소수점 덧셈기, 곱셈기 그리고 나눗셈기 등에 대하여 많은 효과적인 알고리즘들이 발표되었고, 이러한 알고리즘을 기반으로 설계된 연산기들이 모바일을 위한 3차원 그래픽 프로세서에 적용되고 있다. 한편, 3차원 그래픽 처리를 위한 조명처리 엔진과 셰이더 프로세서에는 초월함수 연산기가 이용된다. 초월함수 연산은 사칙연산에 비해 훨씬 많은 연산량을 요구하므로 모바일 환경에 적합한 하드웨어 구조를 설계하기가 더 어렵다.

3차원 그래픽 연산에서 조명효과는 3차원 객체를 구성하는 각 정점의 색상정보를 결정하는 것을 말하며 이는 수식 (1)의 연산을 통하여 얻어진다.[2]

$$i_{tot} = m_{emi} + \sum_{k=1}^{n} c_{spot}^{k} (i_{amb}^{k} + d^{k} (i_{diff}^{k} + i_{spec}^{k})) \quad (1)$$

정점의 최종색상(i_{tot})을 얻기 위해서는 해당 정점에 대하여 i_{amb} (ambient), i_{diff} (diffuse), i_{spec} (specular), c_{spot} (spot-light), 그리고 거리에 따른 감쇠 효과 d (attenuation by distance) 등의 연산을 존재하는 광원의 수인 k번 만큼 반복해야 하기 때문에 조명효과는 광원의 종류 및 그 수에 따라 연산량이 증가되는 특성을 갖고 있다. 이러한 이유로 조명연산 단계는 기하 단계 전체 동작 중에서 가장 많은 연산량을 갖는다.[3]

제곱근과 역제곱근 연산은 이러한 조명연산에서 정점에서 광원을 향하는 단위 벡터와 specular 연산에 사용되는 h 벡터(half vector)를 얻는 연산에 필요하며 셰이더의 기본 연산기로도 사용 가능하다. 일반적으로 제곱근 및 역제곱근의 연산 방법은 반복법과 실함수에 의한 근사법으로 구분할 수 있으며, 반복법에는 직접 계산법(direct methods), Newton Raphson 공식에 기반한 알고리즘, 그리고 정규화 기법(normalization techniques)등이 있다. Newton Raphson 공식은 하드웨어의 복잡도 및 연산량을 증가시키는 나눗셈 연산을 필요로 하기 때문에 모바일 환경의 하드웨어 설계에는 잘 사용되지 않으며, 그림 1에 나타나 있는 직접 계산법의 하나인 Non-restoring 방식이 하드웨어로 구현될 때 가장 많이 사용되고 있다. 그러나 이러한 반복법들은 제곱근 값을 얻기 위하여 반복적인 연산을 수행하

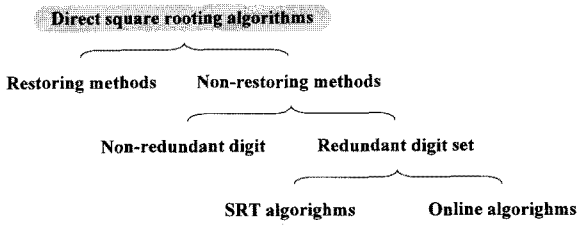


그림 1. 직접 제곱근 연산 알고리즘의 분류
Fig. 1. Direct square rooting algorithms.

므로 하드웨어로 구현될 경우 잠복기(latency)가 증가하는 단점을 가지고 있다.[4~6]

본 논문에서는 이러한 반복적인 연산을 피하고 복잡한 연산을 최소화하면서 OpenGL ES 기반의 3차원 조명효과 연산에 필요한 오차범위를 만족하는 제곱근 및 역제곱근 연산의 하드웨어 구조를 제안한다. 제안하는 방식은 실함수에 의한 근사법으로 가장 잘 알려진 방법인 Taylor 전개식을 기반으로 하며 이를 부동소수점 형식에 맞는 가장 간략한 다항식의 형태로 전개한 수식을 사용한다. 또한 이렇게 전개된 수식은 참조 테이블과 부분선형(piecewise linear) 근사 방식으로 연산이 가능하다. 참조 테이블은 IEEE-754 단정도 (single precision) 및 모바일 환경을 위해 축소된 24 bit 부동소수점 형식에 공통적으로 사용할 수 있다. 제안하는 구조는 참조된 값에 대한 간단한 보정 연산을 수행하여 OpenGL ES 1.x에서 요구하는 10^{-5}의 정밀도를 거의 만족시키면서 참조 테이블의 크기를 줄일 수 있다. 제안하는 제곱근 및 역제곱근 연산의 하드웨어 구조는 축소된 참조 테이블 및 간단한 덧셈, 곱셈 등의 연산으로 구성되기 때문에 하드웨어 복잡도를 최소화할 수 있으며 1 사이클의 잠복기 후에 연산 결과를 얻을 수 있다.

Ⅱ. 연산기 구조

1. 연산 알고리즘

부동소수점 형식은 일반적으로 sign(s), biased exponent(e), fraction(f)의 세 부분으로 구성되어 있으며 수식 (2)는 이러한 구성을 수식으로 표현하고 있고, 그림 2는 IEEE-754 단정도 32 bit 및 축소된 24 bit 부동소수점 형식의 구성을 보이고 있다.[7]

$$x = (-1)^{s} (1.f) 2^{e-bias} \quad (2)$$

수식 (2)에 제곱근과 역제곱근 연산을 적용하면 sign bit가 1일 때는 NaN(Not a number)가 되고 0일 때는

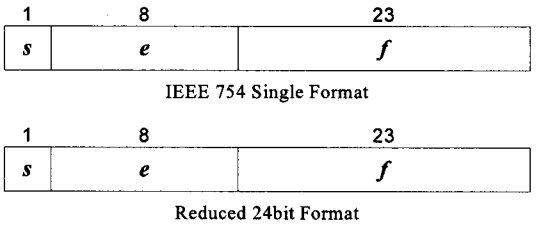


그림 2. 부동소수점 형식의 이진 표현 방식
Fig. 2. Binary representation of floating point numbers.

각각 수식 (3)과 수식 (4)를 얻을 수 있다.

$$\sqrt{x} = 2^{\frac{e-bias}{2}}(1.f)^{\frac{1}{2}} \tag{3}$$

$$\frac{1}{\sqrt{x}} = 2^{-\frac{e-bias}{2}}(1.f)^{-\frac{1}{2}} \tag{4}$$

수식 (3), (4)에서 *e-bias*가 홀수가 될 경우, $\frac{1}{2}$ 연산의 나머지가 발생하여 2의 지수에 소수점 이하의 수가 발생하여 연산이 복잡해지고 최종 결과에 오차가 커지는 결과를 가져온다. 따라서 이를 수식 (5), (6)과 같이 보정하여 연산을 수행한다. 이 경우 지수 부분은 간단한 더하기, 빼기와 논리 쉬프트를 통하여 연산이 가능하다.

$$\sqrt{x} = \begin{cases} 2^{\frac{e-bias-1}{2}}(1.f)^{\frac{1}{2}}2^{\frac{1}{2}}, if\ (e-bias)\ odd \\ 2^{\frac{e-bias}{2}}(1.f)^{\frac{1}{2}}, if\ (e-bias)\ even \end{cases} \tag{5}$$

$$\frac{1}{\sqrt{x}} = \begin{cases} 2^{-\frac{e-bias-1}{2}}(1.f)^{-\frac{1}{2}}2^{\frac{1}{2}}, if\ (e-bias)\ odd \\ 2^{-\frac{e-bias}{2}}(1.f)^{-\frac{1}{2}}, if\ (e-bias)\ even \end{cases} \tag{6}$$

수식 (7)은 Taylor 전개식을 나타내고 있으며, 이는 미분과 나눗셈 연산의 반복으로 하드웨어 구조에 적합하지 않게 보이지만 h 값이 작아질수록 고차항들의 값이 급격히 작아진다는 성질을 이용하면 2차항 이상의 항들을 무시하고 연산이 가능해지므로 한 번의 곱셈과 덧셈만으로 값을 얻을 수 있다.

$$f(x+h) = f(x) + f'(x)h + \frac{f''(x)}{2!}h^2 + \ldots + \frac{f^{(n)}(x)}{n!}h^n + \ldots \tag{7}$$

따라서 $x+h$를 $1+a+b$ $(1>>a>>b)$로 치환하여 b에 대하여1차항까지 전개하면 수식 (8), (9)를 얻을 수 있으며 전개된 연산 수식이 부분선형 근사 방식을 사용하기에 매우 적합한 형태를 보이게 된다.

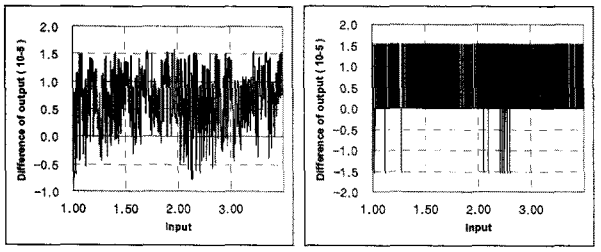
$$\begin{aligned}(1+a+b)^{\frac{1}{2}} &\approx f(1+a) + f'(1+a)\cdot b \\ &= (1+a)^{\frac{1}{2}} + \frac{(1+a)^{-\frac{1}{2}}}{2}b\end{aligned} \tag{8}$$

$$\begin{aligned}(1+a+b)^{-\frac{1}{2}} &\approx f(1+a) + f'(1+a)\cdot b \\ &= (1+a)^{-\frac{1}{2}} - \frac{(1+a)^{-\frac{3}{2}}}{2}b\end{aligned} \tag{9}$$

여기서 $1+a$ 와 관련된 값들은 참조테이블에 저장하고 b와 관련된 값을 연산하여 오차를 보정한다. 참조 테이블에서 b와 관련된 값을 제거함으로써 참조 테이블의 크기를 줄이면서 원하는 오차범위를 만족시킬 수 있다.

2. 오차 분석

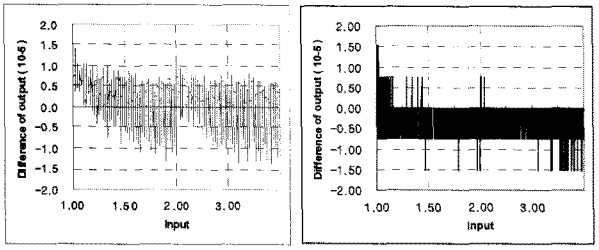
제안하는 구조에 사용된 참조 테이블은 주소 값의 bit 수에 따라 그 정확도가 증가하는 특성을 가지며 실험결과 7 bit의 주소를 갖는 경우 성능 대비 면적의 효율이 가장 좋게 나타났다. 이 경우 24 bit 형식에서 제곱근 연산 결과는 fraction에서 최대 1 bit의 오차를 나타내고, 역제곱근 연산결과는 fraction에서 최대 2 bit의 오차를 갖는다. 동일한 참조테이블을 IEEE-754 단정도 형식에 적용했을 경우에는 오차가 발생하는 bit의 범위는 증가하지만 이 경우 fraction의 값이 더 넓은 범위의 값을 나타내므로 최대 오차의 절대 값은 크게 다르지 않아 여전히 동일한 정밀도를 만족한다. 따라서 동일한 참조테이블의 사용이 가능하다. 그림 3과 4는 각각 제곱근과 역제곱근 연산에 대해 제안된 구조에 따라 설계된 Verilog-HDL 모델과 표준 C의 math 라이브러리 함수의 연산 결과의 차이를 보이고 있다. 오차는 참조 테이블로 계산할 수 있는 모든 값에 대해 이루어졌다. 오차는 참조 테이블로 계산할 수 있는 모든 값에 대해 계산했다. 최대 오차가 1.5 x 10^{-5}으로 나타나고 있는데 이는 참조테이블과 보정연산을 수행한 fraction의 절대 오차이며 OpenGL ES에서 요구하는 10^{-5}의 오차를 거의 만족한다. 참조테이블 크기를 증가시키거나 오차 보정 연산을 추가하여 오차를 더 줄일 수는 있으나 조명 효과 연산에서는 좌표변환 연산처럼 높은 정밀도를 요구하지 않아 오차를 더 줄여도 출력 영상에 영향을 주지 않는다. 따라서 면적 대비 성능 효과가 가장 좋은 현재의 구조를 유지해도 필요한 오차 범위는 충분히 만족한다.



(a) 32bit (b) 24bit

그림 3. 제곱근 연산에 대한 HDL 모델과 C 모델 간의 연산 오차율

Fig. 3. Error rate of square root calculation for HDL model and C model.



(a) 32bit (b) 24bit

그림 4. 역제곱근 연산에 대한 HDL 모델과 C 모델 간의 연산 오차율

Fig. 4. Error rate of inverse square root calculation for HDL model and C model.

III. 설계 및 구현

1. 연산기 구조

그림 5는 제곱근과 역제곱근 연산기의 공통적인 하드웨어 구조를 나타낸다. 입력된 값은 $1+a+b$의 형식으로 표현되는데 IEEE-754 표준의 이진 표현 방식에서 1은 숨겨진 bit이고 fraction에 해당하는 부분을 a와 b로 나누어 1차 참조 테이블과 2차 참조 테이블의 주소 값으로 이용한다. 이때 a와 b를 각각 몇 bit로 결정하느냐에 따라 정밀도와 참조테이블의 크기가 결정된다. a가 커지면 정밀도는 좋아지지만 참조 테이블이 커져 연산기의 크기가 증가하고 b가 커지면 참조 테이블의 크기는 작아지지만 정밀도가 떨어지게 된다. 또한 참조 테이블에 저장된 값의 bit 수에 따라 정밀도가 달라지는데, 그 크기가 클수록 정밀도가 좋아진다. 따라서 fraction a, b의 크기와 참조 테이블에 저장된 값의 bit 수에 따라 원하는 정밀도를 결정할 수 있다. 한편 e-$bias$ 값이 홀수일 때는 참조 테이블에서 얻은 값에 $\sqrt{2}$를 곱해야 한다. 그러나 이러한 연산을 하는 것 보다

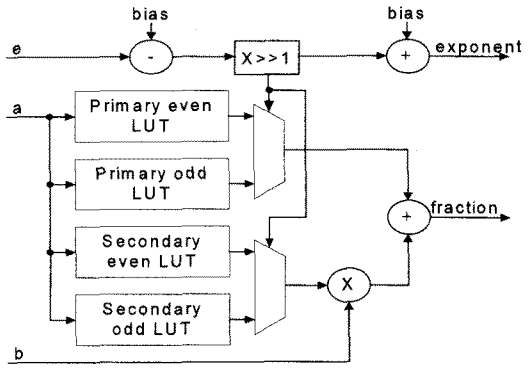


그림 5. 제곱근과 역제곱근 연산을 위한 하드웨어 구조

Fig. 5. Hardware architecture for square root and inverse square root.

는 짝수와 홀수인 경우에 대해 각각 별도의 참조 테이블을 구현함으로서 $\sqrt{2}$를 곱하는 곱셈기를 사용했을 경우에 비해 면적과 연산속도 면에서 이득을 볼 수 있다.

2. 설계 결과

제안된 구조에 따라 Verilog-HDL을 이용하여 제곱근 및 역제곱근 연산기를 설계하였다. 설계된 연산기는 파라미터 값의 수정에 따라 24 bit 와 32 bit 연산이 가능하며 참조 테이블을 사용하여 구조를 최대한 단순화

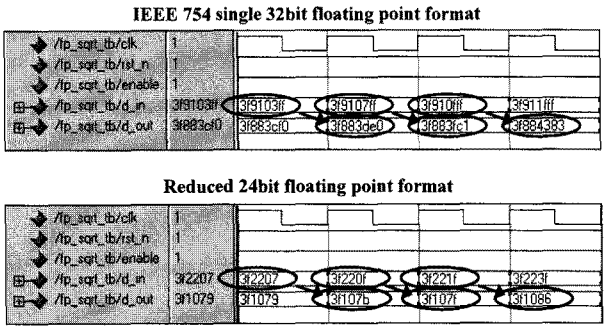


그림 6. 제곱근 연산기의 시뮬레이션 결과

Fig. 6. Result of square root simulation.

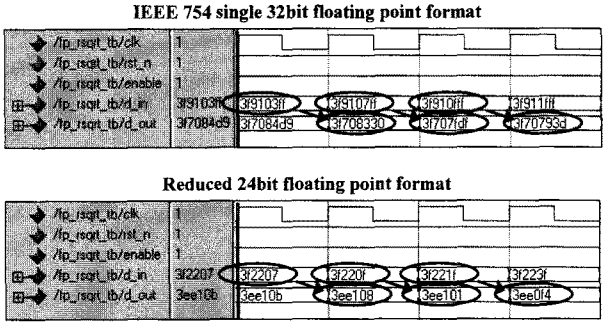


그림 7. 역제곱근 연산기의 시뮬레이션 결과

Fig. 7. Result of inverse square root simulation.

표 1. 0.25㎛ CMOS 공정을 이용한 합성결과
Table 1. Synthesis results using 0.25㎛ CMOS technology.

	IEEE 754		Reduced 24 bit	
	SQRT	ISQRT	SQRT	ISQRT
ROM [bits]	6,528	6,528	6,528	6,528
로직면적 (@100MHz) [gate count]	1,646	1,797	1,063	1,157
최대 동작 주파수 [MHz]	263	263	294	278

표 2. 기존 연산기와의 비교 결과
Table 2. Comparison results with others arithmetic units.

	Latency [cycles]	Throughput [cycle]	Area [mm^2]	Delay [ns]
Square Root				
[9]	15	1	N/A	N/A
[10]	48	1	N/A	N/A
[11]	9	1	N/A	21.7
Proposed	1	1	0.31	3.8
Inverse Square Root				
[4]	5	1	0.41	6.7
[8]	6	1	0.73	7.3
[12]	26	1	N/A	2.4
[13]	2	1	4,000*	6.7
Proposed	1	1	0.32	3.8

* 게이트수

했기 때문에 그림 6과 7의 시뮬레이션 결과와 같이 1 사이클 만에 동작이 가능하다. 이러한 빠른 연산은 조명처리 엔진의 파이프라인을 구성하거나 셰이더의 연산기로 이용될 때 전체 적인 잠복기를 줄일 수 있어 보다 효율적이며 시스템의 성능 향상을 가져올 수 있다.

표 1에는 Synopsys Design Compiler와 0.25㎛ CMOS 셀 라이브러리를 이용하여 설계된 연산기를 합성한 결과가 나타나 있다. 합성은 100MHz 동작 주파수를 기준으로 진행하였으며 참조 테이블의 크기는 6,528 bit이고 로직 면적은 24 bit 연산기가 약 1.100 게이트, 32 bit 연산기는 약 1,700 게이트를 갖는다. 합성된 연산기의 최대 동작 주파수는 263 ~ 294 MHz이다.

표 2에는 IEEE-754 표준의 단정도 부동소수점 형식에 대해 제곱근과 역제곱근 연산기의 성능과 면적에 대하여 기존의 결과와 비교한 내용이 나타나있다. 면적 및 지연시간에 대한 결과는 [13]의 결과를 제외하고는 0.25㎛ CMOS 공정에서 참조 테이블을 포함한 연산기에 대한 결과이다. [13]은 0.18㎛ CMOS 공정에서 합성하였다. 대부분 파이프라인 구조를 이용하여 1 사이클의 throughput 을 보이고 있지만 잠복기는 2 ~ 48 사이클로 제안하는 구조가 다른 논문에 비하여 뛰어난 성능을 보이는 것을 확인 할 수 있다. 또한 면적 면에서도 참조 테이블을 포함한 전체 면적의 크기가 다른 논문들의 결과에 비하여 작으며 지연시간 또한 짧기 때문에 보다 높은 주파수에서 동작이 가능하다. [11]과 [12]는 FPGA에서 동작하는 결과이며 [12]의 경우 동작 주파수가 매우 높지만 잠복기가 26 사이클이다. 제안된 구조도 잠복기를 늘리면 동작 주파수를 더 높일 수 있다. [13]의 경우 로직이 4,000 게이트에 LUT가 3,072 bit인데 제안된 역제곱근 연산기의 경우 로직이 1,157 게이트이고 LUT는 6,528 bit이다. 또한 [13]의 경우 소수점 이하 8 bit까지 정확도를 만족하나 제안된 구조는 15 bit까지 만족하여 정확도면에서도 우수하다. 공정 차이를 고려할 때 잠복기, 면적, 동작 속도, 정확도 면에서 제안된 구조가 더 우수함을 알 수 있다.

3. 동작 검증

설계한 제곱근 및 역제곱근 연산기를 검증하기 위하여 FPGA에 그림 8과 같은 구조의 검증 시스템을 구현하여 검증을 진행하였다. 3차원 그래픽 시스템을 C 언어를 이용하여 구현하고 ARM 프로세서에서 이를 처리하도록 설계하였으며 조명처리 연산 시 필요한 제곱근 및 역제곱근 연산은 설계된 연산기를 이용하여 그 결과값을 얻도록 구성하였다. 즉 소프트웨어적으로 3 차원 연산을 진행하다가 제곱근 및 역제곱근 연산이 필요하면 AMBA AHB를 통해 입력 값을 전달하고 연산된 결과를 받도록 하였다. 그 결과 그림 9와 같이 정상적으로 조명처리가 연산된 영상이 TFT-LCD 화면에 나타나는 것을 확인하였다.

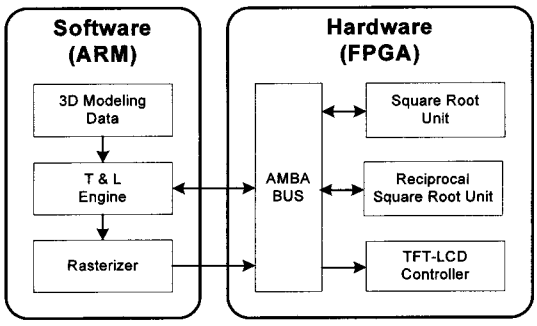


그림 8. FPGA와 ARM 프로세서를 이용한 검증시스템의 블록다이어그램
Fig. 8. Block diagram of verification system using an FPGA and an ARM processor.

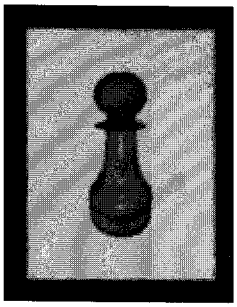


그림 9. FPGA 검증결과
Fig. 9. Result of verification using an FPGA.

Ⅳ. 결 론

본 논문에서는 모바일 3차원 그래픽 프로세서를 위한 제곱근 및 역제곱근 연산기의 하드웨어 구조를 제안하고 이를 설계 및 검증하였다. 제안한 구조는 반복법을 사용하지 않고 Taylor 전개식을 이용하여 참조 테이블과 부분 선형 보간법을 사용하여 참조 테이블 크기를 줄이고 전체적인 동작 속도를 증가시켰으면 OpenGL ES 1.x 에서 요구하는 10^{-5}의 정밀도를 거의 만족시킨다. 또한 이를 Verilog-HDL을 이용하여 24 bit와 32 bit의 부동 소수점 연산이 가능하도록 설계하였다. 설계된 연산기들은 0.25㎛ CMOS 공정에서 합성한 결과 263 ~ 294 MHz에서 동작함을 확인하였고 FPGA와 ARM 프로세서를 이용하여 3차원 그래픽 연산과정에서 정상적으로 동작함을 검증하였다.

참 고 문 헌

[1] "*OpenGL ES Common/Common-Lite Profile Specification Version 1.1.10 (Full Specification)*", Khronos Group Inc., April 4, 2007.

[2] T. Akenine-Moller, E. Haines, "*REAL-TIME RENDERING*", A K PETERS, pp .27-30, p.83, 2002.

[3] J.-h. Sohn, R. Woo and H.-J. Yoo, "*Optimization of Portable System Architecture for Real-Time 3D Graphics*", ISCAS, Vol 1, pp. 769-772, 2002.

[4] M. J Schulte, K. E. Wires, "*High-speed inverse square roots*", 14th IEEE Symposium on Computer Arithmetics, pp. 124-131, April 14-16 1999.

[5] W. Chu, Y. Lim, "*Cost/performance tradeoff of n-select square root implementations*", Computer Architecture Conference, pp. 9 - 16, Feb 2000.

[6] X. Wang, B. E. Nelson, "*Tradeoffs of Designing Floating-Point Division and Square Root on Virtex FPGAs*", 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, pp.195-203, 2003.

[7] IEEE754-1985, *IEEE Standard for Binary Floating Point Arithmetic*, IEEE, 1995.

[8] H. Kwan, R. L. Nelson, and E. E. Swartzlander, Jr.,"*Cascaded Implementation of an Iterative Inverse Square Root Algorithm with Overflow Lookahead*", in Proceedings of the 12th Symposium on Computer Arithmetic, pp. 114-123, 1995.

[9] Y. Li, W. Chu, "*Implementation of Single Precision Floating Point Square Root on FPGAs*", IEEE Symposium on FPGAs for Custom Computing Machines, pp. 226-232, 1997.

[10] Y. Li and W. Chu, "*Parallel-Array Implementations of A Non-Restoring Square Root Algorithm*", International Conference on Computer Design, pp. 690-695, 1997.

[11] "*Floating Point Pipelined Square Root Unit ver 2.07*," Digital Core Design, 2003.

[12] "*Floating Point Inverse Square Root (ALTFP_INV_SQRT): Megafunction User Guide*," Altera, Dec., 2008.

[13] 김정훈, 김기철, "2-Stage Pipeline 구조를 이용한 역제곱근 연산기의 설계," 한국정보과학회 가을 학술발표논문집, vol.34, No.2(B), pp.198-201, 2007.10

저 자 소 개

이 찬 호(정회원)
대한전자공학회 논문지
제43권 SD편 제9호 참조
현재 숭실대학교 정보통신전자공학부 교수