논문 2006-43SD-9-8

Programmable Vertex Shader를 내장한 3차원 그래픽 지오메트리 가속기 설계

(Design of a 3D Graphics Geometry Accelerator using the Programmable Vertex Shader)

하 진 석*, 정 형 기*, 김 상 연*, 이 광 엽**

(Jin-Seok Ha, Hyung-Gi Jeong, Sang-Yeon Kim, and Kwang-Yeob Lee)

요 약

버텍스 쉐이더는 fixed function T&L(Transformn and Lighting) 엔진의 유연성을 향상시키고, 이전보다 다양한 3D 그래픽 효과를 표현하기 위하여 설계되었다. 본 논문의 쉐이더는 DirectX 8.1 의 Vertex Shader 1.1 과 OpenGL ARB 에 기초하여 설계하였다. 버텍스 쉐이더는 벡터 연산을 위하여 4개의 ALU로 구성된다. 작은 면적의 저전력 설계를 위하여 32비트 부동소수점 데이터 형식을 24비트 데이터 형식으로 대체하였다. 버텍스 쉐이더 코어의 동작 검증을 위하여 Xilinx Virtex2 300M gate 모듈을 사용하였다. 시납시스 합성결과 TSMC 0.13um 공정에서 115MHz의 주파수로 동작가능하고, 12.5M Polygons/sec 의 연산성능을 보였다. 버텍스 쉐이더 코어의 면적은 동일 공정에서 11만 게이트를 차지한다.

Abstract

A Vertex Shader is designed to show more 3D graphics expressions, and to increase flexibility of the fixed function T&L (Transform and Lighting) engine. Design of this Shader is based on Vertex Shader 1.1 of DirectX 8.1 and OpenGL ARB. The Vertex Shader consists of four floating point ALUs for vectors operation. The previous 32bits floating point data type is replaced to 24bits floating point data type in order to design the Vertex Shader that consume low-power and occupy small area. A Xilinx Virtex2 300M gate module is used to verify behaviour of the core. The result of Synopsys synthesis shows that the proposed Vertex Shader performs 115MHz speed at the TSMC 0.13um process and it can operate as the rate of 12.5M Polygons/sec. It shows the complexity of 110,000 gates in the same process.

Keywords: Vertex Shader, Geometry processor, OpenGL, 3D Graphics

I. 서 론

최근에는 휴대용 정보기기에서 3D 그래픽의 활용이 핵심기술로 부각되고 있으며, 일부 제품에서는 3D 그래픽 기술을 이용한 게임 및 컨텐츠와 같은 응용서비스를 시작하였다. 그러나 휴대용 정보기기의 한정된 자원과 환경으로 인하여 3D 그래픽의 표현에 기술적인 한계가 있다. 이러한 기술적인 한계를 극복하기 위해서 CPU 프로그래밍 기반으로 3D 그래픽을 가속하여 사용하였다[1]. 초기의 3D 가속기는 단순히 Rendering 만을 처리하였고 Geometry 는 CPU를 이용하여 처리하였다. 3D 그래픽 처리를 위해서는 많은 데이터를 연산해야 하며 그 연산 과정 또한 복잡하다[2]. 보다 정교하고 현실감 있는 3D를 표현하기 위해서는 데이터 양이 기하급수적으로 증가하게 된다. 모바일 환경의 CPU만으로 이를 처리 하기에는 부담이 크기 때문에 3D 그래픽 처리를 위한 하드웨어 가속기에 대한 연구가 이루어졌으며, 꾸준히 발전하여 왔다. 3D 그래픽 처리 과정은 정점의 좌표를 변환하고 광원의 효과를 반영하여 정점의 색을 결

* 학생회원, ** 정회원, 서경대학교 컴퓨터공학과
(Department of Computer Engineering, Seokyeong University)

※ 본 논문은 서울시 SoC 혁신 클러스터 육성 지원사업의 지원으로 작성되었으며 IDEC의 지원장비를 활용하였습니다.

접수일자: 2006년7월1일, 수정완료일: 2006년8월18일

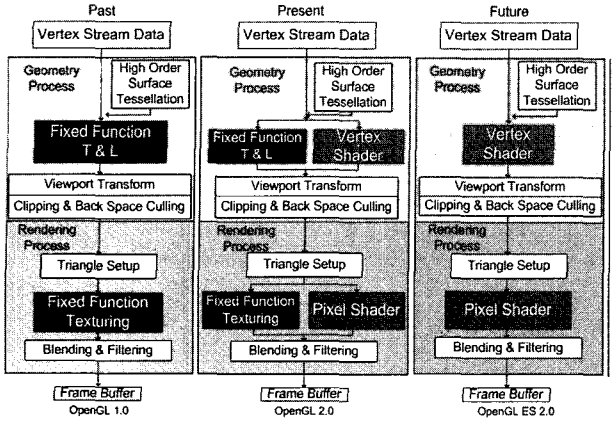

그림 1. 3차원 Graphics 프로세서 구조의 발전 동향
Fig. 1. Trend of 3D Graphics Processor Architecture.

정하는 Geometry 단계와 픽셀의 색과 좌표를 정하는 Rendering 단계로 구분된다. 과거의 3D 그래픽 가속기 구조는 Fig.1과 같이 fixed function T&L(Transform & Lighting) 엔진으로 Geometry 처리를 하였고, fixed function Texturing을 사용하여 Rendering 을 하였다. 현재는 Geometry 처리를 fixed function T&L (Transform & Lighting) 엔진과 Vertex Shader를 병렬로 배치하여 선택적으로 사용하고 있다. Fixed function T&L(Transform & Lighting) 엔진으로 처리할 수 없는 기능들을 Vertex Shader를 이용하여 처리 하거나, 독자적으로 처리를 한다. 향후에는 Vertex Shader만을 사용하여 3D 그래픽 처리를 하게 될 전망이다.

본 논문에서는 휴대 정보기기 시스템에서 더욱 향상된 실시간 3D 그래픽 가속 능력을 갖는 그래픽 시스템의 SoC (System on Chip) 구현을 위해 효과적인 Geometry 처리 구조를 연구하였다. 이를 기반으로 3D 그래픽 Geometry 처리과정에서 3D 그래픽 표현에 프로그래머의 다양한 프로그래밍을 가능하게 하는 Vertex Shader 프로세서를 설계하였다.

II. 본 론

1. 3D 그래픽 Geometry 처리 과정

3D 그래픽 데이터를 처리하는 과정은 Fig.2 와 같이 Application Stage, Geometry Stage, Setup Stage, Rasterization Stage 로 구분된다.

가. Fixed T&L Engine

Geometry 처리는 호스트에 의해 처리된 모델 데이터의 정보들을 입력받으면서 시작된다. 모델 데이터는 한

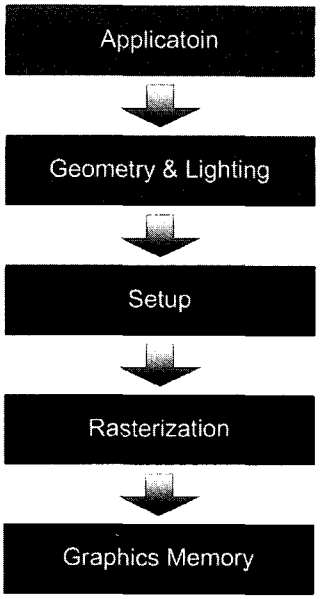

그림 2. 3D 그래픽 파이프라인
Fig. 2. 3D Graphics Pipeline

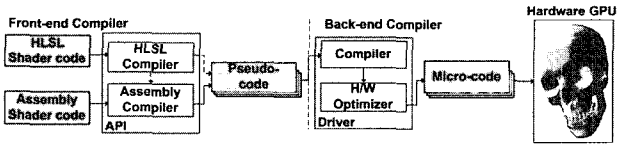

그림 3. API & Shader Interface
Fig. 3. API & Shader Interface.

좌표의 정보, 각 좌표의 수직 방향을 나타내는 Normal Vector, 물질과 광원의 Ambient, Diffuse, Specula 색 속성 정보와 각 단계에서 요구되는 파라메터들이 포함된다. Transformation 처리 과정에서는 모델의 좌표를 시점과 이동 정보로 변환하는 Model/view Transformation 과정, 3D View Volume 으로 투영하는 Projection 과정, View Volume 밖에 위치하는 모델의 정점을 절단하는 Clipping 과정, 모델의 정점 좌표 x, y, z를 w로 나누는 Divide by W 과정과 2D 화면 좌표로 모델 좌표를 변환하는 ViewPort Transformation 과정으로 구성된다. Lighting 처리 과정에서는 모델을 구성하는 정점의 수직 방향을 나타내는 노말 벡터의 변화, 광원에 대한 반사율을 계산하기 위한 방향 벡터인 L, S 벡터 계산, 광원의 감쇠 효과 계산을 위한 광원과 모델의 정점 간의 거리 D 계산, 집중 조명 광원 처리를 위한 Spot, effect 계산과 이를 이용하여 모델의 정점 색 R, G, B를 계산하는 과정으로 구성된다. 광원이 어떻게 물체의 재질의 매개 변수들과 상호작용을 하는지를 결정하고, 결과적으로 화면상에서 특정 물체가 점유하고 있는 정점들의 색상을 결정한다[3].

나. Vertex Shader

프로그래밍 가능한 3D Geometry 처리 표준은 DirectX의 Vertex Shader와 OpenGL의 Vertex Program에서 각각 Macro Assembly 레벨의 표준을 제안하고 있다. 이러한 표준에 의해 만들어진 결과물은 기존의 fixed function T&L (Transform & Lighting) 보다 훨씬 뛰어난 그래픽 효과를 보이고 있다. Programmable Vertex Shader는 fixed functoin T&L (Transform & Lighting) 엔진의 진화형으로 좌표 변환 처리와 광원 처리는 이미 결정된 연산을 수행하게 된다. 변환 처리에 프로그래머가 수정을 가할 수 있는 구조가 Programmable Vertex Shader이다. Programmable Vertex Shader는 정점 좌표의 연산만이 아니라 광원 처리에 있어서도 활용 가능성이 넓어진다. 광원처리는 3개의 벡터를 통해 계산된다.

2. Programmable Vertex Shader 구조 연구 및 설계

제안하는 Vertex Shader 는 DirectX 8.0 vertex Shader 1.1과 OpenGL ARB[4]를 기준으로 설계하였다. Geometry 연산 처리를 프로그래밍할 수 있게 하였으며, 다양한 3D 그래픽 벡터 연산을 위해서 4개의 floating point 연산을 동시에 처리할 수 있도록 SIMD 구조로 이루어져 있다. 본 논문에서는 Fig.3과 같이 OpenGL의 API를 사용하여 생성된 Shader 명령어를 FPGA에 프로그램된 Vertex Shader 로 처리하는 구조로 설계하였다.

Vertex Shader 명령어는 Fig.4 와 같이 최대 3단계의 연산과정을 거치며 Macro 명령어 LIT, POW 를 제외한 모든 명령어는 기본적으로 1 Instruction / 1 cycle 로 처리 되며, Special Function 명령어 2^n(EXP, EX2), Log2n(Ex2, LG2), 1/N(RCP), 1/sqrt(n)(RSQ) 도 모두 1 cycle 에 처리되어 고속의 floating point SIMD연산을 가능하게 한다[5,6].

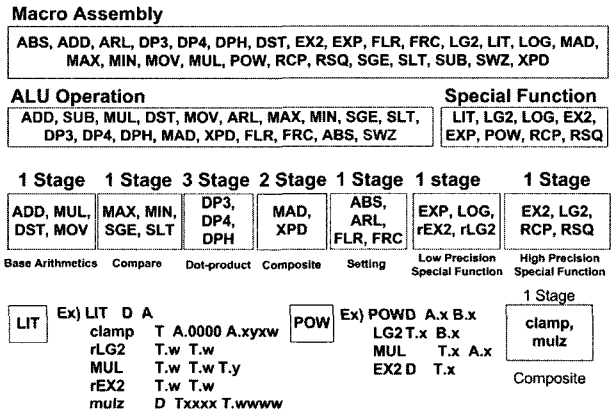

그림 4. 명령어 분석

Fig. 4. Instruction Analysis.

Programmable Vertex Shader는 OpenGL 2.0 ARB를 기반으로 3D 그래픽 처리를 하고 있다. 그렇지만 본 설계는 Mobile 환경을 타겟으로 하고 있기 때문에 작은 LCD, Mobile 자체의 제한적인 성능 등에 의해 그 표현 능력은 OpenGL 표준에 비해 저하될 수 밖에 없다. 따라서 Fig.5 처럼 32bit의 데이터 포맷을 24bit의 데이터 포맷으로 축소하여 적용하였을 경우에 시각적인 차이가 거의 없기 때문에 24비트의 데이터 포맷을 사용하였다.

레지스터는 DirectX의 Vertex Shader1.1과 OpenGL ARB[6]를 지원하고 있으며, Core의 융통성을 위하여 내부 Temporary Register가 4개 추가되었다. 명령어의 추가 및 확장이 가능한 Instruction Set 구조를 설계하기 위하여 MO-LUT (Micro-Operation Look-Up Table)을 사용하였다. MO-LUT는 총 64개의 LUT로 구성되어 있다. 각 레지스터는 명령어의 ALU에서 수행할 세부 동작을 정의하는 Micro-Operation을 저장한다. Micro- Operation은 소프트웨어를 이용하여 변형이 가능하게 하였다.

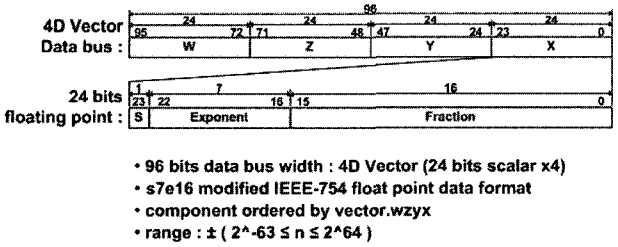

그림 5. 데이터 구조

Fig. 5. Data Organization.

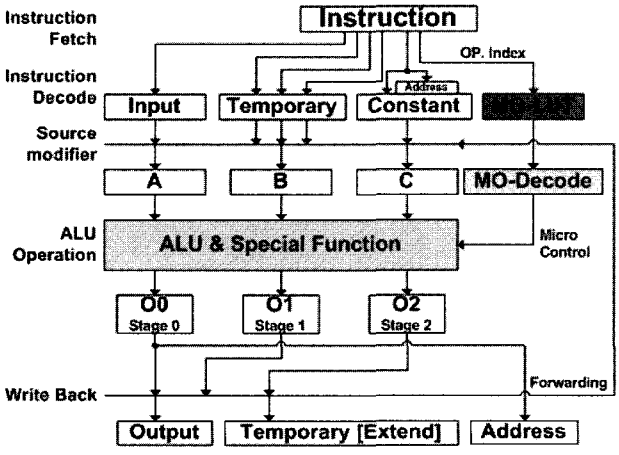

그림 6. Micro-Operation 흐름도

Fig. 6. Micro-Operation Flow Diagram.

API SDK Assembly Compatibility

	OpenGL ARB	Current Shader Instructions	DirectX Vertex Shader 1.1	
matched	ABS	ABS		Absolute Value
	ADD	ADD	ADD	Add
	ARL	ARL		Address Register Load
	DP3	DP3	DP3	Tree-component Dot Product
	DP4	DP4	DP4	Four-component Dot Product
	DPH	DPH		Homogeneous Dot Product
	DST	DST	DST	Distance Vector
	EX2	EX2	EXP	Exponential Base 2
	EXP	EXP	EXPP	Exponentiate (approximate)
	FLR	FLR		Floor
	FRC	FRC	FRC	Fraction
	LG2	LG2	LOG	Logarithm Base 2
	*LIT		LIT*	Light Coefficients
	LOG	LOG	LOGP	Logarithm Base 2(approximate)
	MAD	MAD	MAD	Multiply and Add
	MAX	MAX	MAX	Maximum
	MIN	MIN	MIN	Minimum
	MOV	MOV	MOV	Move
	MUL	MUL	MUL	Multiply
	*POW			Exponentiate
	RCP	RCP	RCP	Reciprocal
	RSQ	RSQ	RSQ	Reciprocal Square Root
	SGE	SGE	SGE	Set On Greater or Equal Than
	SLT	SLT	SLT	Set On Less Than
	**SUB		SUB**	Subtract
	SWZ	SWZ		Extended Swizzle
	XPD	XPD		Cross Product
unmatched		rEX2		Exponential Base 2(Rough)
		rLG2		Logarithm Base 2(Rough)
		clamp		Clamp
		mulz		Multiply on Z
		NOP	NOP	No Operation

* : macro Instruction
** : convertible Instruction

그림 7. 어셈블리 레벨 명령어의 호환성
Fig. 7. Primitive Assembly Level Compatibility.

제안하는 Vertex Shader 프로세서는 OpenGL ARB 의 명령어와 DirectX 의 Vertex Shader 1.1 명령어에 모두 호환하도록 28개의 Primitives 명령어와 2개의 Macro 명령어로 구성하였다. 28개의 명령어를 실행 Cycle 수에 따라 ALU Operation 명령어와 Special Function 명령어로 구분하였다. 명령어는 MO-LUT의 Index를 포함하며, 해당하는 MO-LUT 내용에 따라 Vertex shader의 구체적인 동작 방식을 정의하게 된다. 이 방법은 명령어가 다양한 연산의 조합으로 이루어지는 Core에 적합하며 향후 소프트웨어를 이용하여 Core의 기능을 추가할 수 있도록 한다. 본 논문의 쉐이더에서 사용할 수 있는 명령어는 Fig.7 에서 확인할 수 있다.

III. 검증 시스템 및 결과

Vertex Shader를 검증하기 위해 Fig.8 과 Fig.9 에 보인 검증 플랫폼을 자체적으로 제작하였다. 플랫폼은 FPGA 에 프로그램 된 H/W 모듈과 소프트웨어로 PC 에 구현된 S/W 에뮬레이션 모듈, Test를 위한 3D 모델

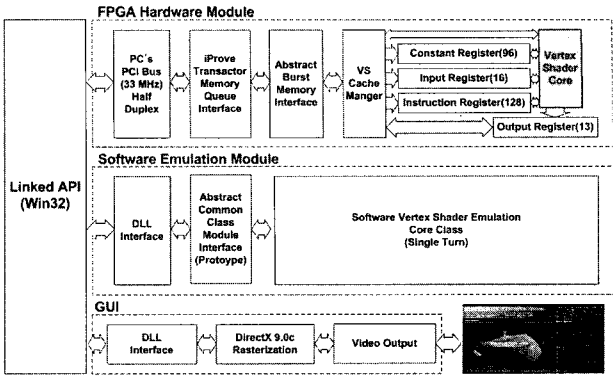

그림 8. Verification Platform
Fig. 8. Verification Platform.

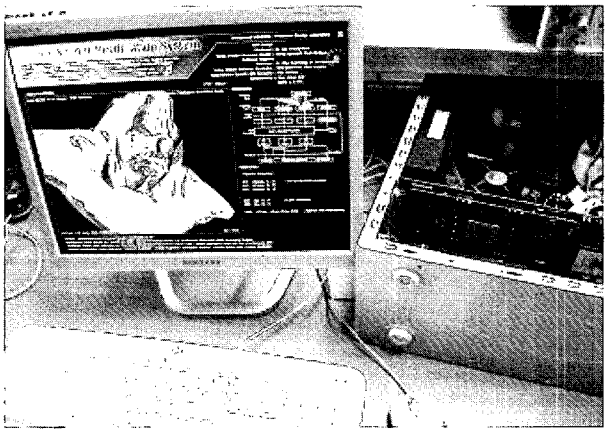

그림 9. 플랫폼 데모 화면
Fig. 9. Platform View.

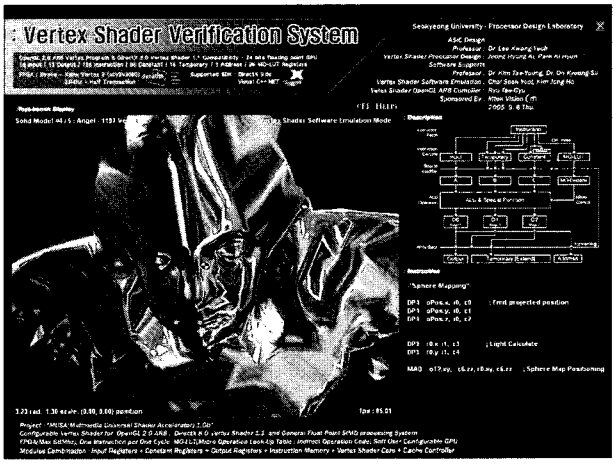

그림 10. DirectX GUI
Fig. 10. DirectX GUI.

을 제공하는 API 와 GUI로 구성된다. S/W 에뮬레이션 모듈과 H/W 모듈의 결과를 비교할 수 있으며, API를 통해 다양한 모델을 검증할 수 있다. GUI는 검증 컨텐츠와 결과에 대한 다양한 정보를 제공한다.

Vertex Shader의 Hardware 모듈과 Software 에뮬레이션 모듈을 이용한 그래픽 화면을 선택적으로 출력하

표 1. 3D Graphic 가속기 프로세서 크기 비교
Table 1. Size comparison of the 3D Graphics accelerator.

3D Graphic Accelerator Processor	Gate
Instruction base T&L Engine[5]	160K
Hardwaired base T&L Engine[9]	32K
Fixed-Point SIMD Vertex Shader[10]	230K
Proposed Vertex Shader	111K

표 2. 3D Graphic 가속기 성능 비교
Table 2. Performance comparison of the 3D Graphics accelerator.

3D Graphic Accelerator	Process Speed
Instruction base T&L Engine[5]	4 Mpolygons/sec @ 80MHz
Hardwaired base T&L Engine[9]	8.9 Mpolygons/sec @ 80MHz
Fixed-Point SIMD Vertex Shader[10]	7.2 Mpolygons/sec @400MHz
Proposed Vertex Shader	12.5 Mpolygons/sec @ 115MHz

여 시각적인 비교를 할 수 있도록 하였다. Hardware 모듈과 Software 에뮬레이션 모듈의 인터페이스는 동일한 Prototype 으로 구성하였기 때문에 동일한 입출력을 가진다. Shader의 연산 결과는 Fig.10과 같이 DirectX 로 구현된 GUI에 전달하여 화면에 출력하였다.

Vertex Shader core 의 size 는 약 11만 게이트이며, Instruction base T&L Engine 보다 작은 면적이다. 처리속도는 최고 12.5M/sec의 polygon 이다.

Ⅳ. 결 론

3차원 그래픽 쉐이더 명령어 기반의 그래픽 가속기를 Fully Hardware로 설계하였으며 명령어는 OpenGL ARB 및 DirectX 8.0과 호환성을 유지하였다.

설계된 Vertex Shader 는 프로그램이 용이한 범용 SIMD Floating point Processor의 특징을 가지고 있다. MO-LUT의 설정이 3D Geometry 처리에 적합하게 구성되어 있어 컨텐츠 개발에 유용하다. 설계된 Vertex Shader Core 의 면적은 약 11만 게이트이고, 최대 12.5M Polygons / 115Mhz로 동작하며 40만개 이상의 Vertex 를 30fps 로 처리 가능하며 개발되고 있는 Fragment Shader와 더불어 완전한 쉐이더 기반의 3차원 그래픽 가속기가 기대된다.

참 고 문 헌

[1] Makoto Awaga, Tatsushi Ohtsuka, Hideki Yoshizawa and Shigeru Sasaki, "3D Graphics Processor Chip Set," IEEE Micro, pp.37-41. dec., 1995.

[2] J. C. Jeong, W. C. Park, W. Jeong, T. D. Han, M. K. Lee "A Cost-Effective Pipelined Divider with a small Lookup Table", IEEE Transactions on Computers, pp.489-495. 2004.

[3] Tomas Akenine-Moller, Eric Hainess "REAL-TIME RENDERING", Second Edition, AKPeters Natick, Massachusetts, 2003.

[4] James C. Lelterman. Learn Vertex and Pixel Shader Programming with DirectX® 9. Wordware Publishing, Inc.

[5] 김재우. 모바일 3D 그래픽을 위한 음영처리 프로세서 설계 (서경대학교 대학원2004년 12월).

[6] 박용진. 역함수를 이용한 작은 룩업테이블을 갖는 조명처리 지수 연산 유닛. (연세대학교 대학원 2004년 12월).

[7] Microsoft MSDN Vertex Shader 1.1 (http://msdn.microsoft.com/archive/default.asp?url=/archive/en-us/directx9_c/directx/graphics/reference/Shaders/ VertexShader1_1/vertex.asp)

[8] Woo Kyeong Jeong, "A SIMD-DSP/FPU for High- Performance Embedded Micro processors," Phd Thesis, Yonsei University, Dec, 2002.

[9] Inseok Song, JinSeok ha, MyungHwan Kim, KwangYeob Lee, TaeHyun Jo, "A Design of a Verification System for a Hardwired 3D Graphics Geometry Engine", The 13th Korean Conference on Semiconductors, pp.887-888, 2006.

[10] Juho Sohn, Ramchan Woo, Hoijun Yoo, "A Programmable Vertex Shader with Fixed-Point SIMD Datapath for Low Power Wireless Applications", The Eurographics Association 2004, pp. 107-104

저 자 소 개

하 진 석(학생회원)
2001년 2월 서경대학교
컴퓨터과학과 학사.
2003년 2월 서경대학교
컴퓨터공학과 석사.
2003년~현재 서경대학교
컴퓨터공학과 박사과정.
<주관심분야 : 저전력 마이크로프로세서, Computer Arithmetic, 암호프로세서, Embedded System, 3D Graphics System >

김 상 연(학생회원)
2006년 서경대학교
컴퓨터공학과 학사.
2006년~현재 서경대학교
컴퓨터공학과 석사 과정.
<주관심분야 : 저전력 마이크로프로세서, 3D Grphics, GPU, SoC>

정 형 기(학생회원)
2005년 2월 서경대학교
컴퓨터공학과 학사
2005년~현재 서경대학교
컴퓨터공학과 석사과정
<주관심분야 : VLSI CAD, Software Design, SoC, 3D Shader>

이 광 엽(정회원)
1985년 8월 서강대학교
전자공학과 학사.
1987년 8월 연세대학교
전자공학과 석사.
1994년 2월 연세대학교
전자공학과 박사.
1989년~1995년 현대전자 선임연구원
1995년~현재 서경대학교 컴퓨터공학과 부교수.
<주관심분야 : 마이크로프로세서, 암호프로세서, Embedded System, 3D Graphics System>