ISSN: 1226-7244 (Print)
ISSN: 2288-243X (Online)
논문번호 16-01-10

j.inst.Korean.electr.electron.eng.Vol.20,No.1,075~081,March 2016
http://dx.doi.org/10.7471/ikeee.2016.20.1.075

타일 기반 그래픽 파이프라인 구조를 사용한 SIMT 구조 GP-GPU 설계

Design of a SIMT architecture GP-GPU Using Tile based on Graphic Pipeline Structure

김 도 현*, 김 치 용*★
Do-Hyun Kim*, Chi-Yong Kim*★

Abstract

This paper proposes a design of the tile based on graphic pipeline to improve the graphic application performance in SIMT based GP-GPU. The proposed Tile based on graphics pipeline avoids unnecessary graphic processing operation, and processes the rasterization step in parallel. The massive data processing in parallel through SIMT architecture improve the computational performance, thereby improving the 3D graphic pipeline performance. The more vertex data of 3D model, the higher performance. The proposed structure was confirmed to improve processing performance of up to 3 times from about 1.18 times as compared to 'RAMP' and previous studies.

요 약

본 논문은 SIMT(Single Instruction Multi Thread)구조 GP-GPU(General Purpose Graphic Processing Unit)에서 그래픽 어플리케이션 성능을 향상시키기 위해 타일 기반 그래픽 파이프라인 구조를 제안한다. 타일 기반 그래픽 파이프라인 구조는 병렬적으로 Rasterization 단계를 처리하고, 불필요한 그래픽 처리 연산은 수행하지 않는다. SIMT 구조를 통해 대용량 데이터를 병렬로 처리하여 연산 성능을 향상시켰고, 이는 3D 그래픽 파이프라인 처리의 성능을 향상하였다. 제안하는 구조를 통해 3D 그래픽 어플리케이션을 처리할 때 3D 모델을 구성하는 정점 데이터가 많아질수록 높은 효율을 보인다. 제안하는 구조는 'RAMP'와 기존의 선행 연구를 비교하여 약 1.18배에서 최대 3배까지의 처리 성능 향상을 확인하였다.

Key words : SIMT; tile-based graphic pipeline; Hierarchy structure; Inside outside test; Rasterization;

---

* Telechips Inc.
★ Corresponding author Dept. of Computer Science, Seokyeong University
e-mail : kcy@skuniv.ac.kr Tel : 02-940-7759

※ Acknowledgment
This work was supported by the Industrial Core Technology Development Program (10049192, Development of a smart automotive ADAS SW-SoC for a self-driving car) funded By the Ministry of Trade, industry & Energy and was supported by Seokyeong University in 2014.
Manuscript received Feb 29, 2016; revised Mar 22, 2016 ; accepted Mar 22. 2016

Ⅰ. 서론

최근 컴퓨터 프로세서의 성능이 높아짐으로써 다양하고 복잡한 연산을 하는 어플리케이션이 발전하게 되고 자연스럽게 시각적으로 화려한 효과를 보여주는 3D 그래픽 어플리케이션도 함께 발전하게 되었다. 하지만 순차적으로 처리되는 범용 프로세서는 방대한 데이터를 가진 3D 그래픽 데이터를 처리하기엔 문제가 있었고 이를 해결하기 위해 GPU(Graphic Processor Unit)가 지속해서 발전해왔다. GPU가 발전하면서 방대한 데이터들과 많은 연산량을 병렬적으로 처리하는 것을 이용해 GPU보다 상대적으로 병렬처리에 문제가 있었던 범용 프로세서를 대체하여 단순히 3D 그래픽 데이터만을 처리하는 것이 아닌 범용적인 병렬적인 연산을 처리할 수 있는 GP-GPU(General Purpose Graphic Processor Unit) 형태로 발전하게 되었다.

본 논문은 GP-GPU 형태의 프로세서에서 효율적으로 범용적인 연산과 3D 그래픽 처리를 위해 멀티스레드를 이용한 그래픽 파이프라인을 연구하였다. 그래픽 파이프라인을 처리하기 위해 tile based rasterizer를 통해 interpolation 과정을 거쳤고 그래픽 파이프라인의 전체 과정을 멀티 스레드 방법을 적용하여 처리 성능을 개선하였다.

Ⅱ. 본론

1. SIMT(Single Instruction Multi Thread) 구조 GP-GPU

하나의 명령어를 멀티 스레드를 이용하여 많은 양의 데이터를 가진 3D 그래픽 어플리케이션을 처리하는 것뿐만 아니라 일반 범용 프로세서에서 처리하지 못하는 복잡한 알고리즘이나 연산을 처리하는 SIMT 구조의 GP-GPU가 발전해왔다.[1]

그림 1은 본 논문에서 적용한 SIMT 기반 멀티 코어 프로세서의 내부 구조이다. 내부는 control path와 data path를 분리하여 여러 개의 SP(Stream Processor)를 하나의 control path에서 제어하여 소모되는 slice register와 slice luts에 사용된 자원량을 줄였다. SP 내부는 3개의 ALU와 하나의 load/store unit이 포함되어 있으며 superscalar issue와 dual warp 구조를 통해 하나의 홀수 warp, 짝수 warp 각각 2개의 명령어를 처리, 총 4개의 명령어를 동시에 처리할 수 있다.

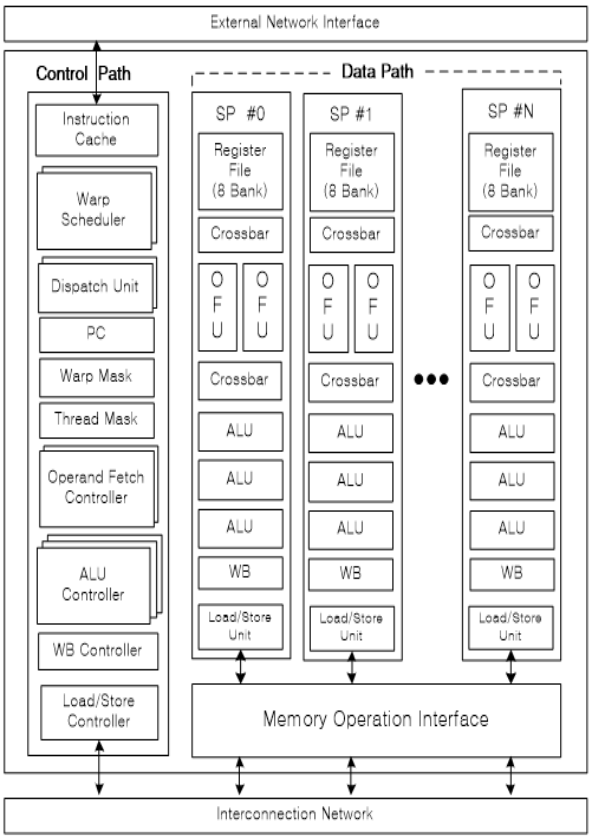

Fig. 1. SIMT based Multi core processor
그림 1. SIMT 기반 멀티 코어 프로세서

2. Instruction

본 논문에서 사용된 SIMT 구조 GP-GPU는 단순히 그래픽 명령만을 수행하는 것이 아닌 프로세서로써 범용적인 연산도 활용할 수 있다. SIMT 구조 GP-GPU는 총 6가지의 명령어 타입을 가지고 있으며 시스템 명령어(SYS), 메모리 명령어(MEM), 데이터 이동 명령어(DATA), 정수형 산술 명령어(INT), 부동소수점 산술 명령어(FLOAT), 확장 명령어(EXTIMM)를 가지고 있다. 해당 명령어 타입 중에 그래픽 연산에 따로 수행되는 그래픽 명령어는 존재하지 않지만, 위에 총 6가지의 명령어 타입들을 조합하여 그래픽 명령어를 대체하여 구현하였다. 표 1은 설명한 명령어의 타입과 세부 기능을 명시하였다.

Table1. Instruction Type
표 1. 명령어 타입

Instruction Type	Description
SYS	System Instructions
MEM	Memory Instructions
DATA	Data transfer Instructions
INT	Integer arithmetic Instructions
FLOAT	Floating point arithmetic Instructions
EXTIMM	Extend immediate

가. 메모리 주소에 따른 명령어

SIMT 구조 GP-GPU에서 사용되는 메모리 명령어 구조는 명령어 코드, 목적지 레지스터, 소스 레지스터, 오프셋으로 구분 돼 있다. 이를 이용하여 메모리 영역을 범용 연산 및 작업등을 수행할 때 사용되는 general memory 영역과 그래픽 관련 연산 및 작업을 수행할 때 사용되는 graphic memory 영역으로 데이터를 Read 하거나 Write를 수행하게 된다.

메모리 명령어를 처리할 때 두 개의 메모리 영역 중 그래픽 메모리 영역일 경우 그래픽 명령어의 동작을 위한 명령어 집합으로 사용하게 되며 그 이외의 메모리 영역을 접근하게 되는 경우 범용 연산 명령어로 판단하고 작업을 수행한다.[2]

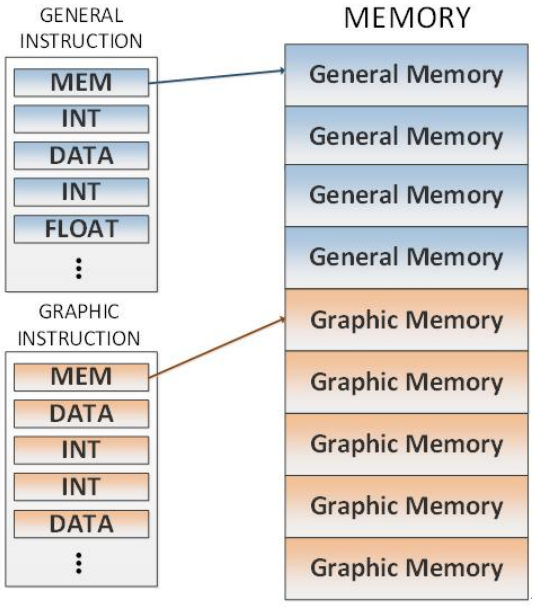

Fig. 2. Type of the instruction on the memory area
그림 2. 메모리 영역에 따른 명령어 분류

나. 명령어 처리 과정

프로세서 명령어는 메모리 맵핑 방식을 사용, 범용 연산 명령어와 그래픽 명령어를 구분하여 사용한다.

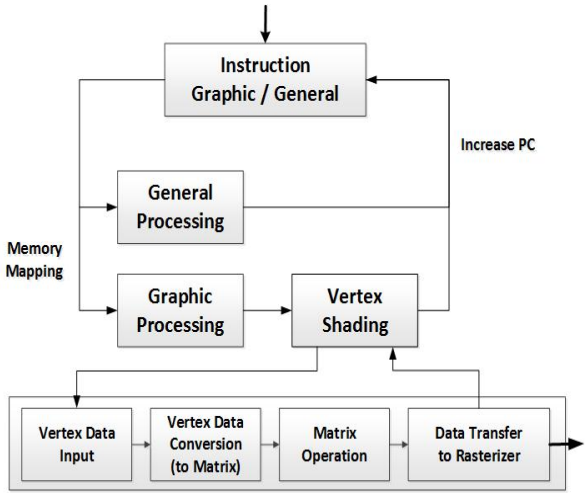

Fig. 3. Instruction processing flow
그림 3. 프로세서 명령어 처리 과정

그림 3과 같이 범용 연산 명령어일 경우 프로세서는 멀티 스레드를 이용하여 대용량의 데이터를 병렬적으로 처리하고 PC(Program Counter)를 증가시킨다. 그래픽 명령어일 경우 프로세서에서 vertex shading 단계를 거치게 된다. vertex shading 단계에서는 많은 양의 정점(Vertex) 데이터를 처리해야 하므로 연산 시간이 매우 증가한다. 하지만 SIMT 구조 GP-GPU는 하나의 명령어를 멀티 스레드를 이용하여 처리하기 때문에 많은 양의 정점 데이터를 효율적으로 처리할 수 있어 프로세서의 지연 시간을 단축 시켰다. 결과적으로 vertex shading 다음 단계인 rasterizer에서 interpolation 단계와 vertex shading 단계에서 발생할 수 있는 성능 저하를 막을 수 있다.

3. Tile based Rasterizer

Tile based rasterizer는 화면을 타일 단위로 나누어 타일마다 복수의 처리기를 두어 연산을 병렬로 처리해 연산 속도를 빠르게 할 수 있는 구조이다. 또한, 타일 단위로 나누었을 때 실제로 연산하는 부분과 연산하지 않는 부분을 나누어 전체 연산량을 줄이는 것이 성능에 큰 영향을 미치게 된다.

본 논문에서는 64*64 크기를 가진 Pixel Block으로 시작하여 각각 16*16, 4*4의 크기를 가진

세 가지의 타일을 사용하였다. 먼저 하나의 폴리곤(Polygon)을 처리하기 위해서는 폴리곤의 내외부를 판정한다.[3]

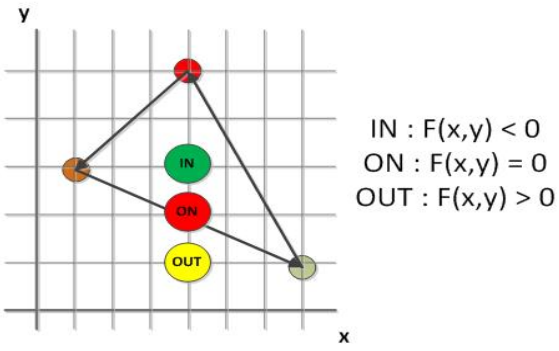

Fig. 4. Inside, Outside Test
그림 4. 폴리곤의 내외부 판정

폴리곤의 내외부 판정이 끝나면 폴리곤의 외부일 때 연산을 제외하여 연산량을 줄이고, 폴리곤의 내부부터 rasterization 작업을 수행하게 된다. 제일 큰 크기의 64*64 타일을 이용하여 그림 4와 같이 in 상태인 영역을 처리한다. 이어서 ON 상태인 경우 64*64 타일로 처리하지 못하는 부분이므로 하위 크기의 타일로 처리하게 된다.

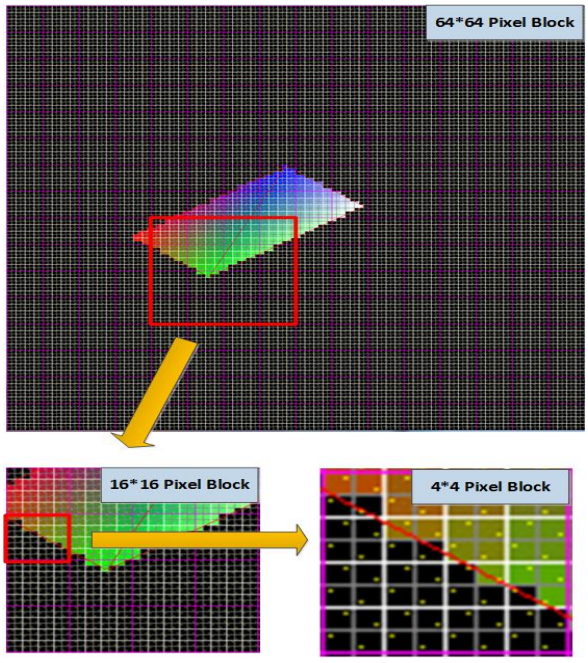

Fig. 5. Hierarchical architecture of the tile
그림 5. 타일의 계층 구조

그림 5는 실제로 tile based on rasterizer를 이용하여 마름모꼴의 사각형을 두 개의 삼각형 폴리곤을 이용해 구현하였다.

4. 화소정보 생성 과정

SIMT 구조 GP-GPU가 그래픽 명령어를 처리할 때 연산 되는 데이터는 메모리의 그래픽 영역에 해당하는 데이터로써 이후 메모리로부터 읽어온 그래픽 데이터를 참조하는 명령어를 그래픽 명령어로 분류하여 vertex shading 과정을 처리한다. vertex shading 이후 수행하는 rasterization 과정은 연산의 종속성으로 인해 연산 속도의 저하를 가져오기 때문에 그래픽 파이프라인의 모든 단계를 프로세서로 처리할 경우 그래픽 명령에 따른 프로세서의 병목현상이 발생한다. 병목현상에 의한 프로세서의 성능 저하를 막기 위해 rasterization을 가속화할 수 있는 그래픽 처리 특화된 장치가 필요하게 되었고 본 논문에서는 그림 6과 같은 처리 과정을 가지는 rasterizer를 설계하여 프로세서의 병목현상을 해결하였다.

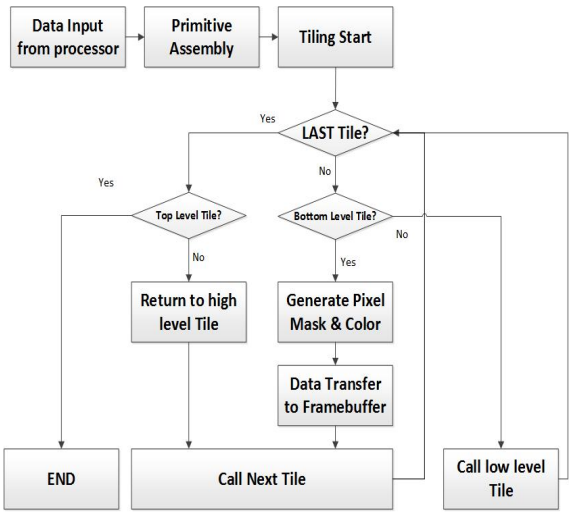

Fig. 6. Process of generating pixel information
그림 6. 화소 정보 생성 과정

Rasterizer는 SIMT 구조 GP-GPU로부터 전달받은 Vertex Shading 처리 후의 정점 정보를 사용하여 Tiling과 Masking, Interpolation을 통해 화면에 출력될 화소 정보의 생성을 가속화하고 메모리의 그래픽 영역 중 출력을 담당하는 프레임버퍼 영역에 저장한다.

설계한 rasterizer의 동작으로 그래픽 파이프라인의 후반 단계인 rasterization을 처리할 수 있으므로 프로세서는 vertex shading의 결과를 rasterizer로 전달하는 것으로 그래픽 처리 과정을 마치고 다음 명령어에 대한 처리를 수행할 수 있다. 본 논문에서는 SIMT 구조 GP-GPU와 rasterizer를 사용한 그래픽 파이프라인의 처리 구조를 통해 그래픽 명령으로 인한 프로세서의 병목현상을 해결하였다.

5. 실험 및 결과

본 논문은 제안하는 구조인 타일 기반 그래픽 파이프라인 구조를 적용한 SIMT 구조 GP-GPU를 검증하였다. 검증에는 Verilog HDL로 구현된 SIMT 구조 GP-GPU와 타일 기반 그래픽 파이프라인 구조가 적용된 rasterizer를 탑재하여 구현하였다. 합성을 진행한 IP는 Virtex-7 XC7VX485T FPGA가 탑재된 Xilinx VC-707 FPGA Board를 사용하여 범용 연산 동작과 그래픽 파이프라인의 처리 과정을 검증하였다. FPGA에 탑재한 IP는 50MHz로 동작한다.

VC707에서 SIMT 구조 GP-GPU와 rasterizer를 동작시키기 위해 AXI4 Bus Interface를 이용하여 주변 IP들과 연동하여 검증 시스템을 구성하였다. 표 2는 FPGA 검증 시스템의 자원 사용량이다.

Table2. Resource usage of FPGA verification system
표 2. FPGA 검증 시스템의 자원 사용량

	Used	Available	Utilization
Slice Registers	137,723	607,200	22%
Slice LUTs	241,631	303,600	79%
Block RAM/FIFO	150	3,090	5%
Number of DSP48E1s	423	2,800	15%
BUFG/BUFGCTRLs	12	32	37%

프로세서는 SDRAM에 저장된 명령어에 의해 동작을 수행하는 도중 메모리 명령어가 그래픽 영역을 참조할 경우 이후의 명령어를 그래픽 명령어로 분류, vertex shading 과정을 수행한다. vertex shading 과정이 완료되면 프로세서는 데이터를 rasterizer로 전달하고 다음 명령어에 대한 처리를 수행한다. 이와 동시에 rasterizer는 그래픽 파이프라인의 후반 단계인 rasterization을 수행하며 연산이 모두 완료되면 생성된 화소 정보를 그래픽 메모리 영역 중 framebuffer 영역에 저장하는 것으로 그래픽 처리를 끝낸다.

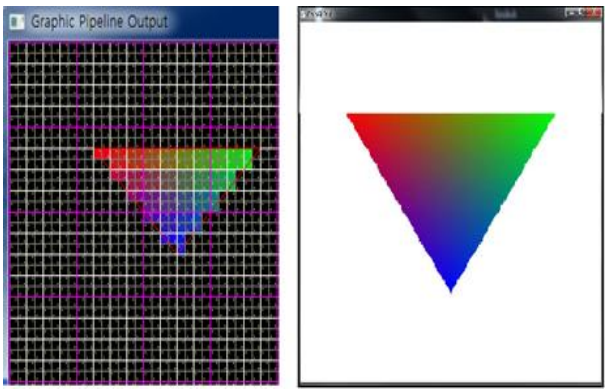

Fig. 7. Comparison of the output result
그림 7. 출력 결과 비교

그림 7의 좌측은 본 논문에서 설계한 GP-GPU가 그래픽 파이프라인을 처리한 화소 정보를 출력한 결과이며, 우측은 OpenGL API를 사용하여 동일한 연산을 진행한 결과이다. 좌측의 경우 하나의 격자는 하나의 tile을 의미하며 본 논문에서 설계한 GP-GPU가 정점 정보와 색상 값을 연산하여 출력한 결과와 OpenGL을 이용한 결과와 매우 유사한 결과 값을 얻은 것을 확인할 수 있다.

표 3은 vertex 개수에 따라 범용 연산이 많은 vertex shading 과정과 그래픽 연산인 rasterizaton 과정의 처리 시간을 비교한 표이다. 실험 (A)의 비해 (B)의 vertex 개수가 약 5배로 증가하였으나 범용 연산에 해당하는 vertex shading 과정은 rasterization 단계보다 처리 시간이 증가하지 않았다. 이는 SIMT 구조를 적용하여 단일 명령어를 멀티 스레드로 처리하여 정점의 수가 늘어나도 처리 시간은 많이 늘어나지 않은 것으로 분석되어 진다.

Table3. Generation time of the pixel information
표 3. 화소 정보 생성 시간

	(A)	(B)
Vertex count	2,946 EA	15,201 EA
Vertex shading	4,187 ns	9,322 ns
Rasterization	5,384 ns	48,104 ns

표 4는 본 논문에서 제시한 rasterizer와 기존에 연구들과 비교한 결과이다. 각각의 기준 동작 주파수가 다르므로 'RAMP'의 동작 주파수인 10MHz 환경에서 동작시켰을 때 QVGA 해상도에서 표 3과 같은 처리 성능을 비교할 수 있었다.

Table4. Comparison - Pixel fillrate per second
표 4. 초당 화소 처리 성능 비교

Rasterizer	Pixel fillrate per second
Proposed Rasterizer	63MPixel/sec
KAIST 'RAMP'[4][5]	21MPixel/sec
Mobile GPU Rasterizer[6]	53MPixel/sec

표 4에서 기존의 연구들과 비교하여 각각 3배, 1.18배의 처리성능 향상을 확인하였다. 또한 본 논문의 그래픽 파이프라인의 경우 50MHz의 동작 주파수에서 최대 316MPixel/sec의 성능을 낼 수 있다.

III 결론

본 논문에서는 SIMT 구조 GP-GPU와 타일 기반 그래픽 파이프라인 구조를 가진 rasterizer를 설계하여 전체 성능을 향상시키기 위해 연구하였다. 추가로 메모리 영역에 따른 명령어를 구분하여 명령어의 추가를 하지 않아 더욱 하드웨어 구조를 가지게 되었다. 범용적인 연산이 많은 vertex shading 과정을 SIMT 구조 GP-GPU가 처리하여 많은 양의 그래픽 데이터가 늘어나더라도 아주 낮은 비율로 처리 시간이 증가하는 것을 확인하였다. 두 번째로 타일 기반 그래픽 파이프라인 구조의 rasterizer를 연구하여 상위 계층에서 불필요한 하위 계층의 호출을 방지하여 불필요한 연산량을 줄였고, 병목 현상을 방지해 전체적인 그래픽 처리 과정의 성능을 높일 수 있었다. 실험 결과 타일 기반 그래픽 파이프라인 구조를 지닌 rasterizer는 선행 연구와 비교하여 약 1.18배에서 최대 3배까지의 처리 성능 향상을 확인하였다.

현재 많은 GP-GPU들은 범용적인 연산과 3D 그래픽 처리를 위해 본 논문 외에도 다양한 연구가 진행되고 있다. 더욱 상호 연구가 진행되어 높은 성능을 보일 수 있는 구조에 관한 결과를 얻길 기대한다.

References

[1] Laszlo, E., "Methods to utilize SIMT and SIMD instruction level parallelism in tridiagonal solvers," Cellular Nanoscale Networks and their Applications (CNNA), 2014 14th International Workshop on, 2014.07

[2] Pinto, C., "GPGPU-Accelerated Parallel and Fast Simulation of Thousand-Core Platforms, Cluster," Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on, pp.53-62, 2011.05

[3] W.F.P.W. Burgers "Tile-Based Rendering," Master's thesis. Technische Universiteit Eindhoven, Eindhoven, 2005.1

[4] Ramchan Woo, "Design and Implementation of Low-Power 3D Graphics SoC for Mobile Multimedia Applications," KAIST, 2004.6.

[5] Jang-seo Ku, "Design of a Rasterizer based on Parallel Processing Interpolation Algorithm for a Mobile GPU," The Graduate School of Seokyeong University, 2013.2

[6] Kim, Sung Su, "Table-based thread reconvergence mechanism on SIMT processor," The Graduate School of Yonsei University, 2011.12

[7] Jeong-Ho Woo, "Mobile 3D Graphics SoC From Algorithm to Chip," WILEY, 2010

[8] Woo-Young Kim, "A Design of a Shader based on the Variable-Length Instruction for a Mobile GP-GPU," The Graduate School of Seokyeong University, 2010.2

BIOGRAPHY

Do-Hyun Kim (Member)

2014 : BS degree in Computer Engineering, Seokyeong University.
2016 : MS degree in Electronics and Computer Engineering, Seokyeong University.
2015 ~ present : Telechips Inc.

Chi-Yong Kim (Member)

1981 : Kyungpook National University, Dept. of Statics(BS)
1986 : Seoul National University, Dept. of Computing Science & Statistics(MS)
1993 : Seoul National University, Dept. of Computing Science & Statistics(Ph. D)
1995 ~ : Seokyeong Univeristy, Dept. of Computer Science, Professor
<Research Interest> Network Traffic Control, QoS, Realtime Scheduling, Embedded System