Sort-Last 병렬 렌더링을 위한 효과적인 메모리 프로세서 구조

윤덕기, 김경수, 이경호, 박우찬
세종대학교 컴퓨터공학과
e-mail:{dkyoon, kskim, khlee}@rayman.sejong.ac.kr
pwchan@sejong.ac.kr

A Processor Architecture with Effective Memory System for Sort-Last Parallel Rendering

Duk-Ki Yoon, Kyoung-So Kim, Kyung-Ho Lee, Wo-Chan Park
Dept of Computer Enginering, Sejong University

요 약

본 논문에서는 각각의 그래픽 가속기에 픽셀 캐시를 사용가능 하게 하면서 성능을 증가시키고 일관성 문제를 해결하는 병렬 렌더링 프로세서를 제안한다. 제안하는 구조에서는 픽셀 캐시 미스에 의한 latency를 감소시켰다. 이러한 2가지 성과를 위하여 현재의 새로운 픽셀 캐시 구조에 효과적인 메모리 구조를 포함시켰다. 실험 결과는 제안하는 구조가 16개 이상의 레스터라이저에서 거의 선형적으로 속도 향상을 가져옴을 보여준다..

1. 서론

근래 들어 비교적 저렴한 가격의 고성능 그래픽 칩들이 발표되어 대부분의 PC에 장착이 되고 있다. 또한, Sony사의 PlayStation®2 [1] 나 MS사의 X-BOX 등의 게임 콘솔 부분에서도 전용 3차원 그래픽 프로세서가 장착되고 있다. 그런데, 현재의 대부분의 3차원 그래픽 프로세서는 다수개의 픽셀 파이프라인을 사용하여 한 개의 삼각형을 고속으로 렌더링 처리하는 구조를 가지고 있다. 이러한 성능은 아주 현실감 있는 영상을 생성할만한 규모의 3차원 데이터를 처리하기에는 여전히 한계가 있다.

최근 반도체 기술의 발전으로 인하여 다수개의 가속기를 내장한 병렬 3차원 그래픽 프로세서가 등장이 가능하게 되었다. [2]에서는 PlayStation®2에 상응하는 Graphics Processing Unit (GPU) 16개와 256-Mb 내장 DRAM을 한 개의 칩에서 구현한 GScube를 발표하였다. 그런데, 16개의 GPU의 출력을 한 개의 pixel merging IC에서 합쳐서 비디오 디스플레이로 보내기 때문에 다수개의 프레임 메모리를 필요로 한다. 이는 [3]에서 규정된 병렬 렌더링 분류 중 sort-last와 유사하다. 이로 인하여 GScube에서는 큰 내장 DRAM을 사용하였다.

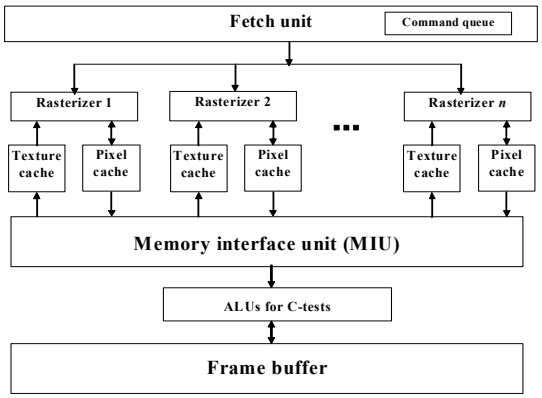
※ "이 논문은 2004년도 한국학술진흥재단의 지원에 의하여 연구되었음."(KRF-2004-041-D00560)

본 논문에서 제안하는 방식은 각각의 가속기 당 픽셀 캐시를 가지고 레스터라이저를 수행하는 새로운 병렬 렌더링 프로세서를 제안한다. 우리는 각각의 픽셀 캐시 일관성 문제를 허락하나 우리는 모든 픽셀 캐시 블락에 consistency-test(C-test)를 추가하여 프레임 버퍼로 쓰여질 때는 일관성을 유지시켰다. 또한 제안하는 구조는 memory interface unit(MIU)로 픽셀 캐시 미스가 발생된 캐시 블락을 전송한 후에 즉시 레스터라이제이션 파이프라인을 실행시킴으로서 latency를 줄일수 있다.

본 논문의 구성은 다음과 같다. 2장에서 우리의 제안하는 구조를 살펴볼 것이다. 3장에서는 제안하는 구조에서의3가지 메모리 시스템에 대해서 논의 할

것이고 4장에서는 시뮬레이션 결과와 성능 평가가 주어질것이다. 5장에서는 결론으로써 향후 계획을 논하였다.

2. 제안하는 구조



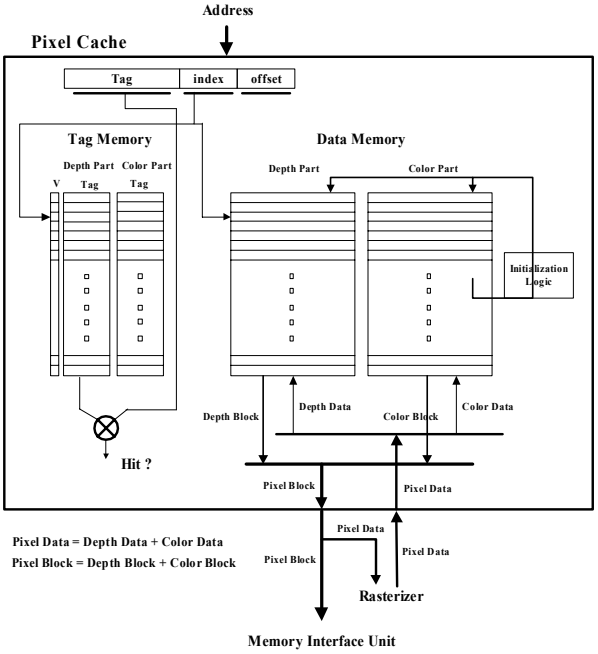
(그림 1) 제안하는 구조

그림 1는 제안하는 병렬 3차원 그래픽 프로세서의 전체 블록 도이다. MIU와 프레임 버퍼 사이에 C-test를 위한 ALU가 삽입 되었다. 그리고 각각의 래스터라이저 당 지역적 픽셀캐시가 장착되었다.

이 논문의 가장 주된 아이디어는 우리는 각각의 픽셀 캐시에서는 일관성 문제가 발생하는 것을 허용하나 프레임 버퍼에서는 일관성을 유지하는 것이다. 제안하는 구조의 프레임 버퍼 일관성 문제는 픽셀캐시에 있는 캐시 블락이 프레임 버퍼로 전송될 때 추가적인 C-test를 수행함으로 유지한다. 제안하는 구조의 다른 아이디어로는 캐시 블록을 전송한 후에 즉시 레스터라이제이션 단계를 수행하는 것 이다. 그러므로 프레임 버퍼로부터 픽셀 캐시로 전송될 응답 블락의 시간을 포함한 캐시 미스에 의한 latency가 감소하게 된다. 더 나아가 레스터라이제이션 파이프라인과 C-test는 독립적으로 수행된다.

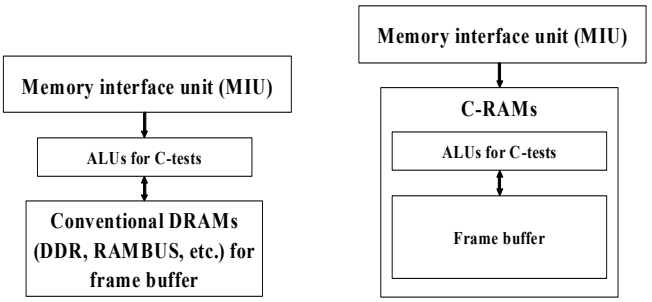
2.1. 제안하는 픽셀 캐시 구조

그림 2에서 제시한 제안하는 픽셀 캐시 구조는 valid bit (V) 부, 깊이 부분 태그 및 색깔 태그로 구성된 태그 메모리, 깊이 데이터 부분과 색깔 데이터 부분으로 구성된 데이터 메모리, 초기화를 위한 초기화 로직으로 구성된다. 일반적으로 픽셀 캐시는 깊이 값이 저장되는 깊이 캐시와 색깔 값이 저장되는 색깔 캐시로 구성된다. 깊이 캐시와 색깔 캐시는 별도로 동작할 수도 있으나, 본 논문에서는 제안하는 알고리즘의 특성상 동일한 스크린 주소를 갖는 깊이 값과 색깔 값이 함께 이동해야 되기 때문에 깊이 캐시와 색깔 캐시는 쌍으로 구성된다. 깊이 값 참조가 색깔 값 참조보다 먼저 수행되기 때문에, 래스터라이저에서의 캐시 참조 시 깊이 값 읽기에 대해서만 태그 비교를 수행하며, 색깔 값에 대한 태그 메모리는 색깔 데이터가 프레임 버퍼로 쓰여지기 위한 주소를 저장하기 위하여 사용된다.

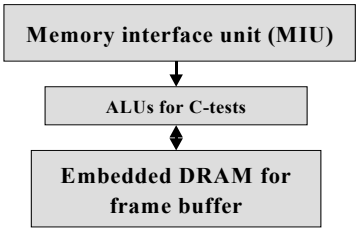


(그림 2) 제안하는 픽셀 캐시 구조

3. 제안하는 메모리 시스템 구조



(a) Conventional DRAM for frame buffer
(b) C-RAM for frame buffer

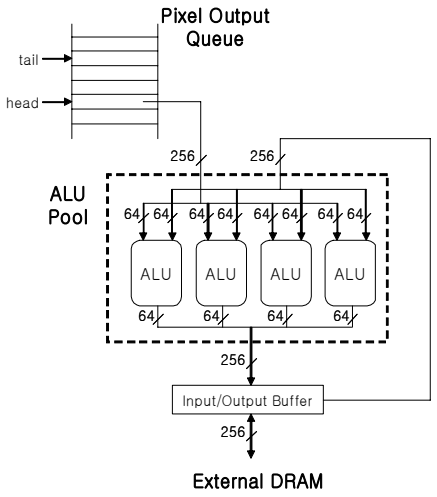


(c) Embedded DRAM for frame buffer

(그림 3) 3가지 메모리 시스템

그림 3은 제안하는 3가지 메모리 시스템을 보여주고 있다. 회색 블락은 렌더링 프로세서 칩 안에 위치하고 있고, 흰색 블락은 칩과 분리되어 있다. 그림 3(a), 프레임 버퍼에 일반적인 DRAM이 사용되었고, 렌더링 프로세서 안에 C-test를 위한 ALU가 포함되어 있다. 다른 그림 3(b) 는 C-RAM이 프레임 버퍼에 사용 되었고 C-test를 위한 ALU가 포함되어 있다. 그림 3(a)에서 프로세서와 프레임버퍼 사이의 관계는 read-modify-write 이고 그림3(b)는 쓰기 전용이다. 그래서 레스터라이제이션 단계에서 프레임 버퍼에 접근 시 그림 3(b) 구조는 그림 3(a)보다 절반의 메모리 대역폭을 요구한다. 그러나 그림 3(a)는 비용 효과적에서 압도적으로 유리하다. 왜냐하면 그림 3(b)의 C-RAM과 같은 새로운 형태의 메모리는 개발하기에 비용이 많이 들기 때문이다.

3,1 메모리 시스템의 내부 구조



(그림 4) 메모리 인터페이스 부의 구성도

그림 4은 메모리 시스템의 블록다이어그램이다 그림 4의 Pixel out queue는 픽셀 캐시에서 보내진 대체 캐시 블록 들을 저장하는 큐이다. ALU pool에서는 C-test(깊이 비교 및 알파 블렌딩)을 수행하는 ALU가 다수 개 존재한다.

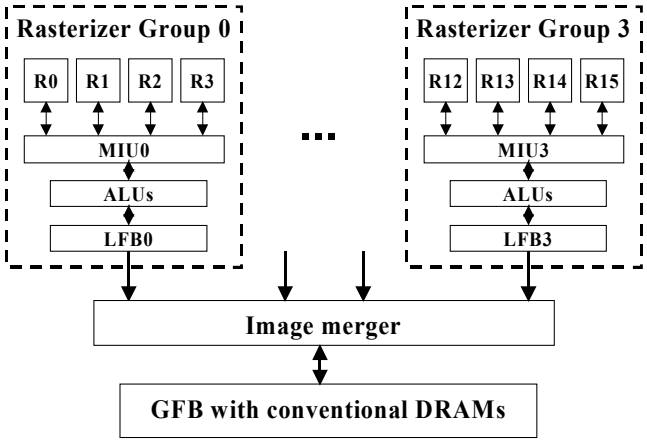
그림 4에서 외부 DRAM과의 데이터 버스는 256 비트라고 하고 픽셀 캐시 블록의 크기는 n×256이라고 가정하면 그림 4의 동작은 다음과 같다. Pixel output queue의 head가 가리키는 대체 캐시 블록에서 처음 256 비트를 ALU pool로 보내고 이와 동시에 같은 주소를 같은 목표 데이터 블록의 256 비트를 외부 메모리로부터 읽어 온다. 256 비트는 4개의 픽셀 정보인 4개의 깊이 값과 4개의 색깔 값으로 구성된다. ALU pool에서는 4개의 픽셀에 대하여 C-test 을 수행한다. 이러한 과정을 거친 4개의 픽셀 데이터는 output buffer를 통하여 외부 메모리에 쓰여진다. 위의 과정을 n 번 수행하면 한 개의 대체 캐시 블록에 대한 연산이 종료된다.

C-test에서 발생되는 n번의 읽기와 쓰기는 burst하게 수행될수 있다. 만약 ALU의 파이프라인이 (n+1) 단계로 구성되어 있다고 가정한다. Burst 읽기 연산과 burst 쓰기 연산 사이에 setup latency가 요구되는데 이 latency를 1이라고 가정한다. 먼저 프레임 버퍼로부터 256비트의 읽기 연산이 n번 수행됨과 동시에 ALU 파이프라인이 실행된다. 그 후C-test를 수행한 후 처음 256비트는 (n+1) 사이클 이후에 출력 버퍼에 나온다. 따라서 마지막 256비트가 나오는데 까지는 (2n+1) 사이클이 걸린다.

4. 실험 시뮬레이션 결과

제안하는 구조의 타당성을 위해서, 다양한 시뮬레이션 결과를 수행하였다. 이를 위하여 Trace-driven 시뮬레이터를 사용되었다. 1600x1200 해상도에서 Mesa Opengl 호환하는 API로 Trace를 3가지 벤치마크(Quake3 demo I, Quake3 demo II, Lightscape)를 사용하여 만들었다. 각각의 벤치마크에서 100 프레임이 각각의 trace를 생성하기 위해 사용되었다. 각각 벤치마크의 모델 데이터는 라운드 로빈 방식으로 레스터라이저에게 분배 되었다. 이러한 trace로 픽셀 캐시 시뮬레이션은 잘 알려진 Dinero III 캐시 시뮬레이터를 변경하여 수행하였다.

4.1 8개 이상의 레스터라이저에서의 성능 향상

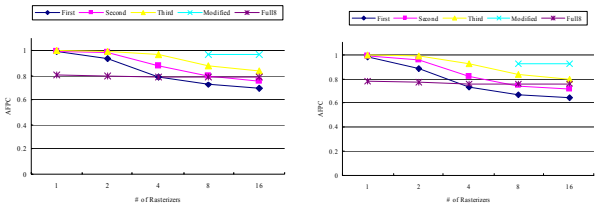


(그림 6) 16개의 레스터라이저를 가지는 변경된 구조

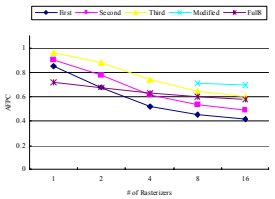
그림 1에서 제안한 구조의 성능은 sort-last 렌더링 시스템에서 scalability를 사용해서 성능 향상시킬수 있다. 그림 6에서 16개의 레스터라이저를 같은 변형된 구조를 보여준다. 레스터라이저가 4개 까지는 효과적인 메모리 시스템을 유지하기 때문에, 4개의 레스터라이져를 1개의 그룹으로 통합하였다. 각각의 그룹이 4개의 레스터라이저를 가지는 제안된 구조는 그림 6와 같다.

그림 6의 전체 구조와 실행 플로우는 sort-last 렌더링 머신과 유사하다. 각각의 레스터라저 그룹은 local frame buffer(LFB)라 불리는 전체 스크린 프레임 버퍼의 작은 이미지를 만들어낸다. 모든 LFB의 내용은 image Merger에 의해서 파이프라인 방식에 따라서 CRT scan 속도로 합성된다. 마지막 합성된 이미지는 global frame buffer(GFB)에 전송된다. LFB와 GFB의 더블 버퍼링을 위해 레스터라이제이션 과 이미지 합성 단계는 중첩되게 실행된다.

4.2 성능 평가



(a) Quake3 I (b) Quake3 II



(c) Lightscape

(그림 7) 제안하는 구조의 AFPCs

분석적인 성능 평가를 위해서 우리는 레스터라이저에서 사이클당 평균 프레그먼트 사이클(AFPC)을 계산하였다. [5]에서는 픽셀 캐시와 텍스처 캐시의 miss penalty가 성능의 감소를 가져온다고 가정하였다. 본 논문에서는 픽셀 캐시에 의한 메모리 latency가 성능 감소를 가져온다고 가정한다. 그러면, AFPC는 다음과 같이 계산된다.

$$AFPC = 1/(1 + Miss\ Rate \times Latency \times (1 - reduction)),$$

Miss Rate 은 픽셀 캐시 미스 비율이고, Latency는 픽셀 캐시 미스에 따른 메모리 대기 시간 사이클 타임이다. 그리고 reduction 은 그림 7에서 보여지는 감소 비율을 나타낸다. 이 방정식의 분모는 레스터라이저에서 사이클당 평균 프레그먼트를 나타낸다. 그림 7는 래스터라이저수와 5개의 다른 메모리 구성에 따른 AFPC를 나타낸다. 0%의 감소비율을 가지고 있는 3번째 메모리 시스템의 AFPC는 픽셀 캐시 미스시 8 사이클이 소용되기 때문에 full8으로 나타내었다. AFPC의 full8은 다른 4개의 제안된 구성과 비교하도록 제공된다. 예를 들어 그림 7 (a)의 4개의 레스터라이저를 보면 AFPC의 full8은 처음 메모리 시스템과 같다. n 개의 레스터라이저에서의 성능 증가는 n 과 AFPC을 곱하면 쉽게 계산된다. 그리하여 그림 7 (a)와 같이 제안된 구조의 AFPC는 16개의 레스터라이저에서 선형으로 증가함을 보인다..

5 결론

이 논문에서 제안하는 새로운 병렬 처리 프로세서 구조는 픽셀 캐시와 픽셀 캐시 미스로 인한 메모리 latency 감소의 consistency 문제를 해결할 수 있다. 실험 시뮬레이션은 제안하는 구조가 16개의 레스터라이저에서 15.5배의 속도 향상을 가져온다. 향후 제안하는 구조의 프로토타입을 만들 계획이다.

참고문헌

[1] M. S. Suzuoki et al., "A microprocessor with a 128-bit CPU, ten floating-point MAC's, four floating-point dividers, and an MPEG-2 decoder," *IEEE Journal of Solid-State Circuits*, vol. 34, pp. 1608-1618, Nov. 1999.

[2] A. K. Khan et al., "A 150-MHz graphics rendering processor with 256-Mb embedded DRAM," *IEEE Journal of Solid-State Circuits*, vol. 36, no. 11, pp. 1775-1783, Nov. 2001.

[3] S. Molnar, M. Cox, M. Ellsworth, and H. Fuchs, "A sorting classification of parallel rendering," *IEEE Computer Graphics and Applications*, vol. 14, no. 4, pp. 23-32, July 1994.

[4] K. Inoue, H. Nakamura, and H. Kawai, "A 10b Frame buffer memory with Z-compare and A-bending units," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 12, pp. 1563-1568, Dec. 1995.

[5] Woo-Chan Park et al., "An effective pixel rasterization pipeline architecture for 3D rendering processors," *IEEE Transactions on Computers*, vol. 52, no. 11, pp. 1501-1508, Nov. 2003.