Mobile 환경에서의 Direct3D 텍스쳐 매핑을 지원하는 효율적인 하드웨어 구조 설계

김상덕[0], 이승기, 박우찬, 한탁돈
연세대학교 대학원 컴퓨터 과학·산업시스템 공학과
askieee@cs.yonsei.ac.kr, {sklee, chan, hantack}@kurene.yonsei.ac.kr

Design of an efficient hardware architecture supporting Direct3D texture mapping in mobile environment

Sang-Duk Kim[0], Seung-Gi Lee, Woo-Chan Park, Tack-Don Han
Department of Computer Science The Graduate School
Yonsei University

요 약

현재 3차원 컴퓨터 그래픽 가속기에서 텍스쳐 매핑과 같은 실감기법을 처리해 주기 위해서는 넓은 대역폭과 많은 메모리를 필요로 한다. 또한 PDA와 같은 차세대 mobile 응용분야에서는 점차적으로 3차원 그래픽의 지원이 요구되고 있는 추세이다. 이를 mobile 환경에서 지원하기 위해서는 낮은 소비 전력 및 적은 메모리, 그리고 하드웨어 비용 등의 제약 요건이 따른다. 그러나 이러한 제약 조건에도 불구하고, mobile 환경에 적합한 3차원 그래픽 하드웨어의 연구는 필수적이다. 본 논문에서는 Windows CE 기반의 mobile 환경에서 Direct3D의 압축 텍스쳐 데이터를 효율적으로 처리하는 하드웨어를 제시한다. 이는 1 cycle에 2개 texel을 처리할 수 있으며, 작은 2-level cache를 사용하여 대역폭을 효과적으로 줄였다.

1. 서 론

3차원 컴퓨터 그래픽은 멀티미디어 환경을 구축하기 위한 가장 핵심적인 연구 분야이다. 특히 텍스쳐 매핑은 실감영상을 생성하는데 있어 핵심적인 기법인데, 이를 처리하기 위해서는 많은 메모리와 넓은 대역폭이 필요하다[1].

지금까지는 이와 같은 3차원 그래픽 환경을 제한적인 부분을 제외하고는 Desktop PC에서만 누릴 수 있었다. 이러한 환경을 mobile 기기 (PDA, 휴대폰 등)에서 구현하기 위해서는 응용기반 설계와 같은 하드웨어에 대한 새로운 접근이 필요하다

Windows CE는 Mobile 기기의 대표적인 OS의 하나이다. 이 OS에서의 3차원 그래픽 라이브러리로 Direct3D를 사용한다[2]. 이는 텍스쳐 데이터 형식으로 압축하지 않은 텍스쳐 데이터와 압축을 한 텍스쳐 데이터를 사용한다. 두 형식 모두 메모리와 텍스쳐 매핑 유닛간의 대역폭을 효율적으로 사용하기 위한 타일 기반 렌더링 (Tile-Based Rendering)을 지원한다[3].

더욱이 mobile환경에서는 메모리 크기와 전력량 그리고 대역폭이 제한되어 있지만, 전체 그래픽 파이프라인에 영향을 미치지 않도록 텍스쳐 매핑 하드웨어가 구현되어야 한다.

따라서 텍스쳐 매핑은 빠른 계산의 요구와 점점 하나의 프레임에 많은 양의 텍스쳐 사용과 mipmaping과 같은 필터링으로 인해 전용 하드웨어를 필요로 한다. 추가적으로 압축된 텍스쳐 데이터를 처리하기 위해서는 일반적인 텍스쳐 데이터 처리보다 디코딩시에 추가적인 계산이 필요하게 된다. 이와 같은 이유로 실시간 렌더링 시스템에서는 텍스쳐 매핑 하드웨어가 반드시 필요하다[4].

본 논문에서는 Windows CE를 기반으로 하는 mobile 환경에서 텍스쳐 매핑 하드웨어를 설계하였다. 이는 Texture Cache와 Decompression Unit 사이의 대역폭을 줄일 수 있는 2-level cache를 사용한다. 또한 이 구조는 Direct3D의 압축 텍스쳐 데이터를 처리하고, bilinear interpolation을 지원한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구를 소개하고, 3장에서는 Direct3D의 압축 텍스쳐에 대해 설명한다. 4장에는 제안하는 하드웨어구조를 제시하고, 제시한 구조의 성능평가를 보여준다. 마지막으로 5장에는 결론과 향후 계획에 대해 서술한다.

2. 관련 연구

Notebook PC를 제외한 대부분의 mobile 기기는 2차원 컴퓨터 그래픽만을 지원한다. 그러나 현재 PowerVR의 MBX 시리즈는 mobile기기에 적합한 3차원 컴퓨터 그래픽을 지원하는 그래픽 하드웨어인데, 이의 성능은 비디오 게임기인 Dreamcast의 성능과 비슷하다[5]. 그리고 텍스쳐를 압축하는 방법에 대한 연구는 BTC (Block

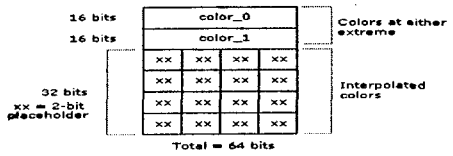
Truncation Coding)를 이용한 것이 일반적이다[6].

3. Direct3D의 압축 텍스쳐 데이터 형식

Direct3D에서는 5가지의 압축 텍스쳐 데이터 형식이 있다. 모든 형식에는 DXT1에서 사용하는 RGB 형식을 가지고 있으며, 추가적으로 alpha값을 가지고 있을 수 있다[7]. 또한 16개의 RGB, alpha값을 하나의 블럭으로 만들어 압축을 하며. 텍스쳐 메모리에서의 지역성을 높였다.

3.1 DXT1

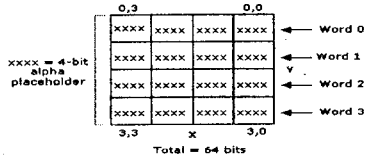
이 압축 형식에는 한 bit의 alpha값을 가지고 있는 형식과 alpha값을 가지고 있지 않은 형식 두 가지가 있으며, 64 bits로 구성되어 있다. [그림 1]은 DXT1의 데이터 형식을 보여준다. 한 블록에는 두개의 대표값(color_0, color_2)을 가지고 있고, 각각의 texel은 이 두 값을 interpolation하여 RGB를 표현한다. 각각의 texel은 2-bit를 가지고 있다.



[그림 1] DXT1에서 RGB 블럭의 데이터 형식

3.2 DXT2, DXT3

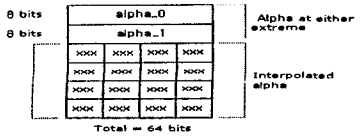
이 형식은 DXT1과 동일한 64-bit의 RGB 블럭을 가지고 있으며, alpha channel에 대한 64-bit 블럭을 추가적으로 가지고 있다. 각각의 alpha는 4 bits를 포함해서 16개의 레벨을 나타낼 수 있다. DXT2는 alpha값이 RGB 값에 미리 곱해져 있는 형식이고, DXT3은 그렇지 않은 형식이다. DXT2, 3의 alpha 블럭은 [그림 2]와 같다.



[그림 2] DXT2, 3에서 alpha 블럭의 데이터 형식

3.3 DXT4, DXT5

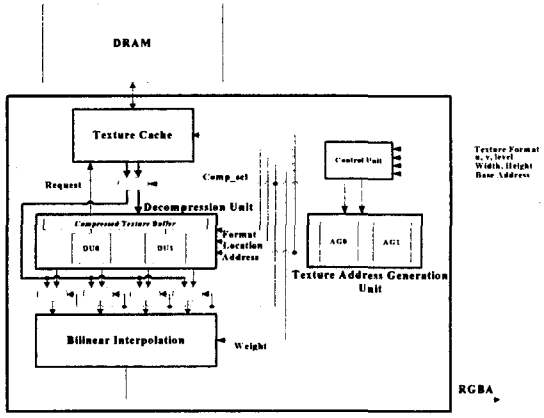
이 형식 역시 DXT1의 64-bit RGB 블럭을 가지고 있고, 추가적으로 64-bit의 alpha 블럭을 가지고 있다. alpha 블럭은 RGB 블록과 마찬가지로 그것을 대표하는 대표값 alpha_0, alpha_1을 가지고 있으며, 각 alpha는 3 bits의 값을 이용해서 대표값을 interpolation하여 총 8개의 alpha값을 구한다. DXT4는 alpha값이 RGB에 미리 곱해져 있는 형식이고, DXT5는 그렇지 않은 형식이다. alpha 블럭은 [그림 3]과 같다.



[그림 3] DXT4, 5에서 alpha 블럭의 데이터 형식

4. 제안하는 하드웨어

제안하는 텍스쳐 매핑 하드웨어 사양은 다음과 같다. DRAM과 Texture Cache의 사이즈는 각각 64MB, 64KB이고, 계산된 영상을 출력하는 부분의 해상도는 최대 800x600으로 가정한다. 그리고 전체 그래픽 파이프라인과 텍스쳐 파이프라인은 1개, 텍스쳐 cache의 output port를 2개로 가정하여, 1 cycle에 2개의 texel이 텍스쳐 매핑 계산에 필요하게 된다. 또한 필터링 방법으로 bilinear interpolation을 사용한다. 이 방법은 하나의 픽셀 색상을 결정하기 위해서 4개의 텍셀이 필요하다. 그러므로 앞선 cycle에서는 4개의 텍셀 중에서 위쪽의 2개의 텍셀을 처리하고, 다음 cycle에서 나머지 아래의 2개의 텍셀을 처리한다.



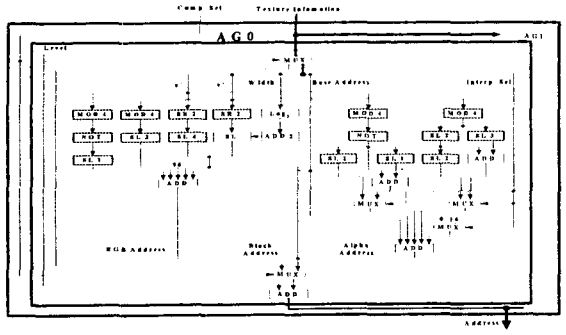
[그림 4] 제안하는 텍스쳐 매핑 하드웨어의 전체 구조

4.1 Texture Address Generation Unit

Texture Address Generation Unit은 1 cycle에 2개 texel의 주소를 계산해야 한다. 따라서 2개의 유닛 (AG0, AG1)이 이것을 구성 하고 있다. 입력으로는 Control Unit에서 u, v 값에 각각 0.5씩을 더하고 뺀 결과를 순차적으로 AG0과 AG1에 보내준다. Comp_Sel은 현재의 텍스쳐가 압축된 형식인지 아닌지를 Texture Format의 입력값으로부터 판단한다. 딸서 이것은 Texture Cache에서 나온 데이터의 Decompression Unit에서 계산 여부를 판별한다. 압축된 텍스쳐인 경우에는 변형된 u, v값과 텍스쳐의 width와 시작주소 등 텍스쳐에 대한 정보가 Texture Address Generation Unit으로 입력된다. 이 정보를 이용하여 RGB와 alpha에 대한 블럭과 그에 대한 offset의 주소를 계산한다[그림 5]. 비압축 텍스쳐의 경우에는 변형된 u, v값이 바로 leve에 대한 정보와 함께 계산된다. 그리고 출력으로 2개 texel의 주소를 포함한 Address가 있다. Address 출력은 Decompression Unit으로 전달된다. [그림 6]에는 내부 구조를 보여준다. 그리고 상대적으로 적은 비용으로 만들 수 있은 모듈은 점선으로 처리하여 일반적인 것과 구분했다.

```
Block Address = base address
                +[(v >> 2) << {(log2(width of  texture array in
                texels) + 2)] +( (u >> 2) <<4  )
RGB Address = Block Address+(optional)
              +(v mod 4) * (bits of RGB) * 4
              +{3- (u mod 4)} * (bits of RGB)
Alpha Address = Block Address + (optional)
              +(v mod 4) * (bits of Alpha) * 4
              +{3- (u mod 4)} * (bits of Alpha)
```

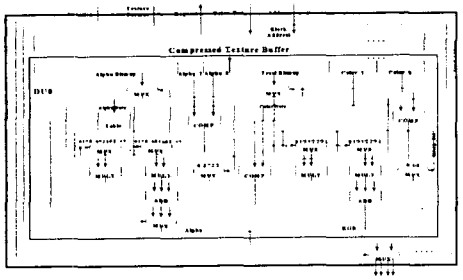
[그림 5] Address 생성식



[그림 6] Texture Address Generation Unit

4.2 Decompression Unit

Texture Address Generation Unit에서 1 cycle에 2개 주소가 입력되므로 2개의 Decompression Unit이 전체를 구성하고 있다. Texture Cache에서 입력으로 들어온 RGB와 alpha 블럭은 16개 texel을 포함하고 있다. 또한 텍스쳐 매핑의 지역성으로 인해 한번 저장된 RGB와 alpha 블럭은 재사용이 가능하다. 또한 사이즈가 큰 Texture Cache에 접근하는 것 보다 적은 사이즈의 버퍼에 접근하는 것이 전력이나 속도에 많은 이점이 있다. 이와 같은 이유로 Compressed Texture Buffer의 데이터를 Decompression Unit에서 재사용이 가능하게 2K bits의 크기를 가지는 Buffer를 추가하였다. 이런 점들이 재사용을 하지 않는 일반적인 텍스쳐 매핑 하드웨어와 크게 다른 점이다. 일단 주소가 Decompression Unit에 입력되면, 가장 먼저 Decompressed Texture Buffer에서 그 주소를 확인한 뒤, 다음 단계로는 Compressed Texture Buffer에서 확인하게 된다.



[그림 7] Decompression Unit

4.3 Bilinear Interpolation

Decompression Unit의 출력을 각각 RGB와 alpha로 나누어 입력을 받아서 계산을 수행한다. bilinear interpolation 방식의 필터링을 사용하기 때문에 4개의 texel을 이용해서 한 픽셀의 텍스쳐 값을 결정한다.

4.4. 성능평가

제안하는 하드웨어의 구조는 1 cycle에 2개의 texel을 계산하여 한 픽셀의 텍스쳐값을 결정한다. 또한 Decompression Unit과 Texture Cache 사이에 또 하나의 cache 역할을 하는 buffer를 두어 데이터의 재사용을 제공하였다. 그러나 기존에 이러한 사양을 지원하는 하드웨어가 존재하지 않기 때문에 직접적인 비교는 불가능하다. 여기서는 제안한 하드웨어에 포함되는 모듈에 대해서만 보여준다[표 1].

[표 1] 제안한 하드웨어에 포함되는 모듈 개수

Unit / Descpription	Texture Address Generation	Decompression Unit
Adder	12 (32bits)	4 (8bits), 4(16bits)
Multiplier	0	8 (8bits), 8 (16bits)
Table	2 (Log)	2 (128 bits)
Multiplexor	5 (2:1)	20 (average 8:1)
Buffer	0	2 (2K bits, 128 bits)

5. 결론

본 논문에서는 모바일 환경에서 필요한 3차원 그래픽 가속기의 한 부분인 텍스쳐 매핑을 효율적으로 지원할 수 있는 하드웨어 구조를 제시하였다. 이는 mobile 기기에서 Direct3D의 텍스쳐 데이터를 효율적으로 처리하는 하드웨어이다. 그리고 제안 구조는 1 cycle에 2개 texel을 처리할 수 있으며, 작은 2-level cache를 사용하여 대역폭을 효과적으로 줄였다.

참고문헌

[1] Kugler A, "High Performance Texture Decompression Hardware," The Visual Computer, Springer, Vol. 13, No.2, p.p. 52-63, 1997.

[2] Microsoft, MSDN, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dx8_c/directx_cpp/Graphics/ProgrammersGu... UsingDirect3D/Textures/Compressed/alphatextures.asp

[3] Ziyad S. Hakura, Anoop Gupta, "The design and analysis of a cache architecture for texture mapping," Proceedings of the 24th international symposium on Computer architecture, 1997.

[4]Kugler A, "Designing a High Performance Texturing Circuit," In Proceedings of IS&T SPIE Symposium on Electronic Imaging Science and Technology, 1997.

[5] PowerVR Technology, http://www.powervr.com/Technology.asp

[6] Delp EJ, Mitchel OR, "Image Compression using Block Truncation Coding," IEEE Transactions on Communication 27(9), p.p. 1335-1342, 1979.