다중접근을 허용하는 3차원 메모리 시스템
(A 3D Memory System Allowing Multi-Access)

이 형 †
(Hyung Lee)

요 약 본 논문에서는 임의의 좌표를 기준으로 17가지 접근방식을 지원하는 3차원 메모리 시스템을 제안한다. 제안하는 메모리 시스템은 메모리 모듈 할당 함수와 주소 할당 함수를 토대로 선 접근방식 13가지, 사각형 접근방식 3가지, 육면체 접근방식 1가지 등 모두 17가지 접근방식을 제공한다. 즉, 임의의 좌표에서 임의의 간격을 갖고 17가지 접근방식 중 어떠한 접근방식 내에서도 다수개의 데이타에 동시접근하는 기능을 제공한다. 이를 위해 제안하는 메모리 시스템은 메모리 모듈 선택 회로, 읽기/쓰기를 위한 데이타 라우팅 회로, 주소 계산 및 라우팅 회로들로 구성된다.

본 논문에서 제안하는 메모리 시스템은 응용 프로그램에 따라 쉽게 확장될 수 있으며, 메모리 시스템에 저장된 데이타를 개발자와 프로그래머가 논리적인 3차원 배열로 간주하여 처리할 수 있도록 데이타의 하드웨어 독립성을 지원한다. 또한 제안한 메모리 시스템은 다양한 접근방식 내의 다수개의 데이타에 동시접근 할 수 있기 때문에 볼륨 렌더링이나 볼륨 클리핑 등과 같은 다양한 3차원 응용 분야 및 다중해상도를 지원하는 프레임 버퍼를 위한 시스템 구조의 메모리 시스템으로써 적합하다.

키워드 : 3차원 메모리 시스템, 다중접근 메모리 시스템, 볼륨 렌더링, 볼륨 클리핑, 다중해상도 프레임 버퍼

Abstract In this paper a 3D memory system that allows 17 access types at an arbitrary position is introduced. The proposed memory system is based on two main functions: memory module assignment function and address assignment function. Based on them, the memory system supports 17 access types: 13 Lines, 3 Rectangles, and 1 Hexahedron. That is, the memory system allows simultaneous access to multiple data in any access types at an arbitrary position with a constant interval. In order to allow 17 access types the memory system consists of memory module selection circuitry, data routing circuitry for READ/WRITE, and address calculation/routing circuitry.

In the point of view of a developer and a programmer, the memory system proposed in this paper supports easy hardware extension according to the applications and both of them to deal with it as a logical three-dimensional array. In addition, multiple data in various access types can be simultaneously accessed with a constant interval. Therefore, the memory system is suitable for building systems related to 3D applications (e.g. volume rendering and volume clipping) and a frame buffer for multi-resolution.

Key words : 3D Memory system, multi-access memory system, volume rendering, volume clipping, and frame buffer for multi-resolution

1. 서 론

영상, 비디오, 그래픽 등의 시각 매체를 처리하는 분야에서는 방대한 데이타를 실시간으로 처리하기 위해 시스템 구성과 확장성 관점에서 많은 연구가 진행되어 왔다. 이들 시각 매체 처리 분야에서는 매체별 특성이 고려된 전용 프로세서의 개발에서부터 시스템 구성의 초기 단계에서 개별 응용 프로그램의 특수성을 고려한 시스템 구현에 이르기까지 실시간 처리를 위한 다양한 접근 방법들이 연구되어 왔다. 이들 연구들은 시스템 복잡도, 다양한 기술과 방법론, 빠른 처리 속도, 그리고 저장 장치 및 입출력 대역폭 등 다양한 세부 분야별로 진행되었다. 또한 시스템 구현에 있어서 다양한 병렬처리 방법들도 함께 고려되었다.

시각 매체를 다루는 응용 분야에서는 프로세서 성능

· 이 논문은 2004년도 한국학술진흥재단의 지원에 의하여 연구되었음 (KRF-2004-003-D00308)
† 정 회 원 : 대전보건대학 방송제작기술과 교수
hyung@hit.ac.kr
논문접수 : 2005년 2월 7일
심사완료 : 2005년 7월 21일

및 메모리 용량의 증가와 더불어 실감형 고해상도 데이타를 실시간으로 처리하기 위한 요구가 증가되고 있다. 그러나 이들 데이타를 실시간으로 처리하기 위해서는 충분한 메모리 대역폭이 확보되어야 한다. 그렇지 못할 경우에는 동시에 처리해야 할 데이타를 충분히 확보하지 못하기 때문에 시스템 성능이 현저하게 낮아지게 된다. 그래서 시각 매체 응용 프로그램을 실시간으로 처리하기 위해 진행된 연구들 중에서 메모리 대역폭을 확보하기 위한 개별적인 연구들이 다양한 형태로 진행되었다[1-3].

실시간 처리를 위해 충분한 메모리 대역폭을 확보하는 기본적인 방법으로는 여러 개의 메모리 모듈을 배열한 인터리빙(interleaving) 방식이 고려되었다. 이러한 방식이 적용된 메모리 시스템은 효율적인 대역폭을 지원하기 위해서 많은 메모리 모듈과 고속 접근이 가능한 메모리 모듈을 사용함으로써 메모리 모듈 개수 만큼의 데이타에 동시접근 할 수 있다. 그러나 이 방식은 단일 접근방식(access type)만을 지원하기 때문에 응용 프로그램에 매우 종속적이다. 즉, 다양한 접근방식이 요구될 경우, 단일 메모리 모듈을 사용하는 것과 같은 시스템 성능을 갖게 된다. 또한 다양한 접근방식이 요구되는 메모리 시스템의 경우, 동시에 동일 메모리 모듈에 접근함으로써 발생될 수 있는 메모리 접근 충돌(access conflict)을 고려해야 한다. 이러한 문제를 해결함으로써 요구되어진 시스템 성능을 유지할 수 있다.

최근들어 특히 3차원 데이타를 실시간으로 처리하기 위한 연구가 이전보다 시스템 구현 측면에서 다양하게 진행되고 있다. 특히 의학영상이나 가상현실 등 3차원 데이타를 실시간으로 처리해야하는 응용 분야에 적용된 시스템은 3차원 데이타를 효율적으로 저장하고 충분한 메모리 대역폭을 지원할 수 있어야 한다. 즉, 메모리 시스템이 시스템 전체의 성능을 크게 좌우하는 중요한 요인으로 간주되고 있다[4-6]. 또한 이들 응용 분야에서 처리하고자 하는 전형적인 데이타 공간 해상도는 512^3 정도로써 방대한 양의 데이타를 포함하고 있다. 그래서 3차원 데이타를 실시간으로 처리하려는 응용 분야에서는 시스템 설계시 병렬처리 기법이 적용된 3차원 메모리 시스템에 대한 연구 및 개발이 필수적으로 진행되어 왔다[7-13]. 이들 시스템에 적용된 3차원 메모리 시스템들은 볼륨 렌더링을 위한 전용 시스템의 메모리 시스템으로써 연구 개발되었기 때문에 어떤 특정 3차원 응용 프로그램 - 예를 들면 볼륨 크리핑 - 에 적용될 경우 데이타를 재배열해야만 하는 오버헤드가 발생하게 된다. 또한 다중해상도(1024^3, 512^3, 256^3 등)를 요구하는 응용 프로그램에 적용될 경우 시스템의 성능이 현저하게 저하될 수 있다.

본 논문에서는 다양한 접근방식을 지원하는 3차원 메모리 시스템을 제안한다. 제안하는 3차원 메모리 시스템은 기본적으로 블록 메모리와 벡터 메모리가 제공하는 접근방식을 제공하며 추가적으로 사각형 형태의 접근방식을 동시에 제공한다. 또한 이들 접근방식 내에서 임의의 간격을 허용한다. 즉, 임의의 좌표에서 임의의 간격을 갖고 직선내의, 육면체내의, 사각형내의 다수개의 데이타에 동시접근할 수 있는 모두 17가지 접근방식을 제공한다. 제안하는 메모리 시스템을 다양한 응용 분야(예를 들면 볼륨 렌더링, 볼륨 크리핑, 다중해상도를 위한 버퍼 등)에 적용할 경우 동일한 시스템 성능을 발휘할 수 있다.

본 논문의 구성은 다음과 같다. 제2장에서는 관련 연구를 통해 3차원 메모리 시스템의 연구 동향을 살펴보고, 제3장에서는 제안하는 메모리 시스템이 제공하는 접근방식을 소개한다. 그리고 제4장에서는 제안하는 메모리 시스템 구성을 기술하고 제5장에서 결론을 맺는다.

2. 관련 연구

방대한 데이타를 실시간으로 처리하기 위해 충분한 메모리 대역폭을 확보하는 일반적인 방법으로는 메모리 모듈(memory module)을 단일 SRAM이나 DRAM 등으로 구성하고, 여러 개의 메모리 모듈들을 하나의 메모리 뱅크(memory bank)로 구성하는 것이다. 즉, 메모리 뱅크는 동일한 입출력 버스를 공유하는 여러 개의 메모리 모듈들로 구성된다. 비록 메모리 뱅크 내의 오직 하나의 메모리 모듈만이 주어진 시간에 메모리 접근 요청에 응답하지만, 이 구조는 인터리빙을 이용해서 단일 메모리 주소에 대해 메모리 접근 주기(memory access cycle) 동안 모든 메모리 모듈들이 응답하게 된다.

인터리빙 기법 중에서 하위(low-order) 인터리빙(그림 1)은 높은 성능을 요구하는 대부분의 컴퓨터 시스템에서 메모리 시스템 구성 기법으로 널리 사용되고 있다. 메모리 모듈의 개수가 2의 제곱인 경우에는, 주소의 하위 몇 개의 비트는 메모리 모듈의 인덱스를 나타내고, 주소의 나머지 상위 비트는 각 메모리 모듈의 주소가 된다. 이러한 방식이 적용된 메모리 시스템이 단일 접근 방식만을 요구하는 특정 응용 분야에 적용되어 최대의 대역폭을 제공하기 때문에 3차원 데이타를 처리하는 볼륨 렌더링과 같은 특정 응용 프로그램을 위한 3차원 메모리 시스템으로 널리 적용되었다. 인터리빙 기법을 적용한 일반적인 3차원 메모리 시스템으로는 블록 메모리(block memory)가 있으며, 동일 메모리 접근 주기에 동시접근 가능한 8개의 데이타를 8개의 메모리 모듈에 저장하기 위한 구성은 그림 2와 같다. 또한 한번의 메모

리 접근 주기 동안 동시접근 가능한 데이타의 개수에 따라 블록 메모리를 4-, 8-way interleaved memory라고 하며, 이들 구조는 VIZARII[9], VOGUS[10], VIRIM[11]의 메모리 시스템에 적용되었다.

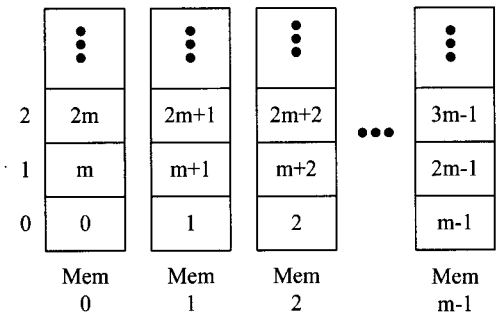

그림 1 m개의 메모리 모듈로 구성된 low-order interleaved 메모리 시스템

이 기법의 최대 단점은 데이타의 접근방식이 순차적일 경우에만 최대의 대역폭을 제공한다는 점이다. 그림 1의 경우를 예로 들면, 데이타의 접근방식이 행인 경우는 한번의 메모리 접근 주기에 m개의 데이타에 동시접근을 허용함으로써 최대의 대역폭을 제공한다. 그러나 접근방식이 열인 경우에는 메모리 접근 충돌이 발생하기 때문에 m개의 데이타에 접근하기 위해서는 m번의 메모리 접근 주기가 필요하다. 또한 전체의 대역폭은 하나의 외부 프로세서에게 종속되기 때문에 여러 개의 프로세서들이 병렬로 데이타를 동시처리하기 위해서는 메모리 뱅크의 복사본이 필요하게 된다.

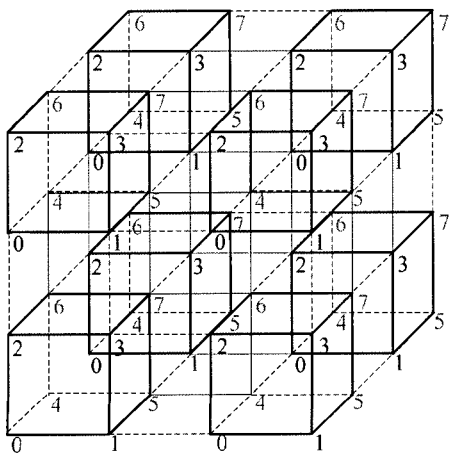

그림 2 8-way 블록메모리내의 메모리 뱅크에 볼륨데이타 할당(숫자는 메모리 뱅크의 인덱스를 나타냄)

3차원 메모리 시스템은 동시접근되는 데이타의 구성형태에 따라 다양하게 설계될 수 있다. 블록 메모리의 경우는 동시접근되는 데이타의 배열형태가 기본적으로 정육면체의 구성을 이루고 있다. 하지만 동시접근되는 데이타의 배열형태가 직선일 경우에는 좀 더 유동적인 인터리빙 정도를 제공하게 된다. 이러한 기능을 제공하는 메모리를 벡터 메모리(vector memory)라고 하며, 높은 데이타 처리능력과 확장성을 제공한다는 특성을 갖는다. 또한 접근방식 내에서 2, 10의 제곱 또는 짝수의 간격을 허용한다. 즉, 블록 메모리는 육면체 접근방식만을 제공하지만 벡터 메모리는 3차원 공간에서 다양한 방향성을 갖는 직선 접근방식을 지원하고, 또한 직선 접근방식 내에서 제한적인 간격을 갖고 데이타에 접근할 수 있다. 3차원 응용 분야 관점에서 볼 때, 벡터 메모리는 블록 메모리보다 좀 더 유연성을 제공한다. 주소 공간을 선형으로 비틀린(linear skewing) 벡터 메모리는 그림 3과 같으며 이 메모리구조를 활용한 볼륨 렌더링 시스템으로는 큐브(CUBE)시스템이 있다[12,13].

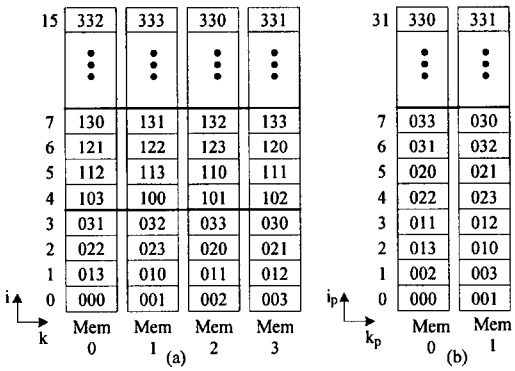

그림 3 4×4×4 볼륨을 위한 3D 비틀림 벡터 메모리 구조. (a) 메모리 모듈 m=4인 경우 (b)메모리 모듈 m=2인 경우. 굵은 라인은 3차원 메모리의 면(slice) 경계를 나타낸다.

상기 3차원 메모리 시스템들은[9-13] 볼륨 렌더링을 위한 전용 시스템의 메모리 시스템으로써 연구 개발되었기 때문에 다른 접근방식을 요구하는 어떤 특정 3차원 응용 분야(볼륨 크리핑, 3차원 다해상도 프레임버퍼 등)를 위한 시스템의 메모리 시스템으로 활용될 경우 앞서 언급한 이유로 인하여 그 시스템의 성능을 저하시킬 수 있다. 즉, 기존에 개발된 3차원 메모리 시스템은 응용 프로그램에 종속적이기 때문에 다양한 3차원 응용분야에 폭 넓게 활용되기 위해서는 좀 더 다양한 접근방식을 제공하면서 특정 응용 프로그램으로부터 적용이 자유로운 3차원 메모리 시스템의 개발이 필요하다.

본 논문에서는 상기 메모리 시스템들[9-13]보다 다양한 접근방식을 제공하는 3차원 메모리 시스템을 제안한다. 제안하는 3차원 메모리 시스템은 기존의 블록 메모리와 벡터 메모리가 제공하는 접근방식을 제공하며 추가적으로 사각형 형태의 접근방식을 동시에 제공한다. 또한 접근방식 내에서 임의의 간격을 허용하는 메모리 시스템이다. 제3장에서는 제안하는 메모리 시스템이 지원하는 접근방식들을 나열하고, 제4장에서는 메모리 시스템 구성 및 기능을 기술한다.

3. 메모리 접근방식

본 논문에서 제안하는 3차원 메모리 시스템은 3가지 분류의 17가지 접근방식를 지원하며, 종류는 선 접근방식 13가지와 사각형 접근방식 3가지 및 육면체 접근방식 1가지이다. 그림 4는 지원하는 접근방식을 나타내며, p,q,r은 제안된 메모리 시스템을 설계함에 있어 주요한 디자인 파라메터이다. 기저좌표(x,y,z)와 접근방식을 지정함으로써 특정 접근방식 내의 pqr개의 데이타에 동시 접근 할 수 있다.

8-way 블록 메모리 구조를 갖는 [10,11]은 그림 4(b)의 *Cube*접근방식($p=q=r=2$인 경우)만을 지원하며, 큐브 시스템[12]에 적용된 벡터 메모리 구조를 갖는 [5]에서는 그림 4(a)와 동일한 선 접근방식을 지원한다. 본 논문에서 제안하는 3차원 메모리 시스템은 이들 접근방식 외에도 그림 4(b)에서 보는 바와 같이 3가지의 사각형 접근방식들(*XYS, YZS, ZXS*)을 추가적으로 지원한다. 사각형 접근방식은 예를 들면 3차원 메모리 시스템에 저장된 볼륨 이미지의 단면을 실시간으로 처리하여 보여줄 수 있는 기능을 제공할 수 있다. 또한 제안하는 메모리 시스템은 기존의 블록 메모리와 벡터 메모리의 장점들을 동시에 지원하기 때문에 높은 유연성을 갖고 다양한 응용 분야에 적용될 수 있다.

4. 3차원 메모리 시스템

3차원 메모리 시스템을 구성하기 위해서는 우선적으로 동시접근 할 수 있는 데이타 개수(n)와 메모리 모듈 개수(m)간의 관계를 설정해야 한다. 일반적으로 $m>n$인 메모리 시스템이 제3장에서 언급했던 모든 17가지 접근방식들 내의 n개의 데이타에 대해 충돌없이 동시접근을 허용한다[14]. 가령, 특정한 접근방식만 허용하는 블록 메모리 구조는 $m=n$인 메모리 시스템으로 구성될 수 있다.

모든 접근방식들 내의 n개의 데이타에 동시접근하기 위해서는 논리적인 3차원 메모리 $L\times M\times N$ 내의 모든 데이타를 물리적으로 구성된 m개의 메모리 모듈들에 할당해야 한다. 논리적인 3차원 메모리 내의 특정 데이타 $I(x,y,z)$는 $0\le x<L,\ 0\le y<M,\ 0\le z<N$ 범위 내에 위치하며, $I(x,y,z)$가 실제로 저장된 물리적 메모리 모듈의 인덱스는 $\mu(x,y,z)\in[0,m-1]$로 표현되고, 해당 모듈 내의 $I(x,y,z)$에 대한 주소는 $\alpha(x,y,z)$로 표현될 수 있다. 메모리 모듈 할당 함수인 $\mu(x,y,z)$가 동시접근 가능한 n개의 데이타를 서로 다른 m개의 메모리 모듈에 할당할 때 주소 할당 함수인 $\alpha(x,y,z)$는 다음을 만족시켜야 충돌없이 n개의 데이타에 동시접근 할 수 있다.

$$\left.\begin{array}{l} I(x1,y1,z1)\neq I(x2,y2,z2) \\ \mu(x1,y1,z1)=\mu(x2,y2,z2) \end{array}\right\} \Rightarrow \alpha(x1,y1,z1)\neq\alpha(x2,y2,z2)$$

4.1 메모리 모듈 할당 함수

메모리 모듈 할당 함수는 데이타가 저장된 메모리 모듈의 인덱스를 결정하는 함수로 다음과 같다.

$$\mu(x,y,z)=(x+yp+zpq)//m$$

여기서 m은 n보다 큰 솟수이며, $x//y$의 연산 결과는 $x\div y$의 나머지로써 음수가 아닌 정수이다. 또한 p,q,r은 디자인 파라메터로써 $pqr=n$이 되며, 이들 디자인 파라메터는 메모리 시스템을 구현함에 있어서 적용될 시스템의 비용과 용량 등 시스템의 성능에 영향을 미치게 된다.

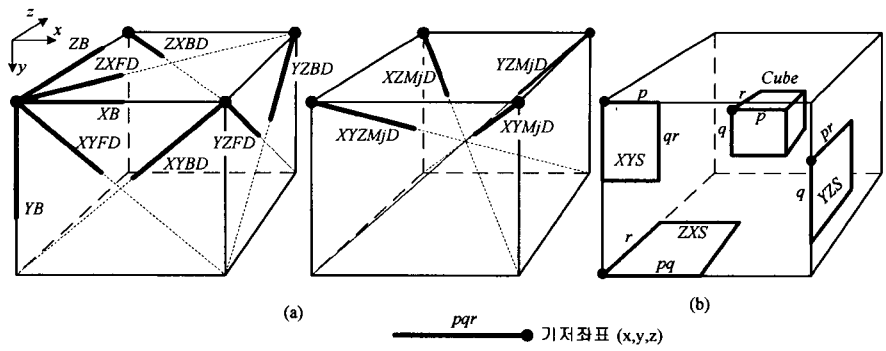

그림 4 제안된 메모리 시스템에서 지원하는 기저좌표(x,y,z)에 따른 다양한 접근방식
(a) 선 접근방식 13가지, (b) 사각형 접근방식 3가지 및 육면체 접근방식 1가지

메모리 모듈 할당 함수를 통해 제3장에서 언급된 각각의 17개 접근방식 내의 n개의 데이타를 m개의 메모리 모듈들로부터 충돌없이 접근 가능함이 [18]에서 증명되었다.

예를 들어 설명한다면, 3차원 메모리 $L\times M\times N$이 8×8×8이고 디자인 파라메터 p,q,r 모두가 2인 경우, 메모리 모듈의 개수(m)는 11, 동시접근 가능한 데이타의 개수는 8이 된다.

4.2 주소 할당 함수

해당 메모리 모듈내에 저장된 데이타의 주소를 계산하기 위해서는 주소 할당 함수가 필요하며 다음과 같다.

$$\alpha(x,y,z)=(y/(qr))\cdot s1+x/p+z\cdot s2$$

여기서 $s1$과 $s2$는 정수로써 각각 ⌈L/p⌉과 ⌈$M/(qr)$⌉ $\cdot s1$의 연산 결과값보다 크거나 같은 정수이며, x/y는 $x\div y$의 몫으로써 음수가 아닌 정수이다. 또한 연산자 ⌈x⌉의 결과는 x보다 같거나 큰 수 중에서 가장 작은 정수이다.

4.1절의 예를 토대로 계산해 본다면, $s1$은 4, $s2$는 8이 된다. $s1$의 의미는 y축의 연속적인 개수들(qr) 간의 차이를 나타내며 $s2$는 z축의 연속적인 값들의 차이를 나타낸다.

4.3 메모리 모듈 선택

m개의 메모리 모듈들로부터 n개의 데이타에 동시접근하기 위해서는 n개의 메모리 모듈을 선택해야 한다. 즉, n개의 데이타에 해당되는 n개의 메모리 모듈들을 활성화시킨다는 것을 의미한다. 그러나 실제 구현에 있어서는 m-n개의 메모리 모듈을 비활성화시키는 것이 회로구현에 있어 좀 더 편리할 것이다.

본 논문에서 제안하는 메모리 시스템은 n개의 데이타에 동시접근하기 위해서 기저좌표(x,y,z)와 접근방식($type$)을 지정해 주어야 한다. 그리고 동시접근될 n개의 데이타에 해당하는 n개의 메모리 모듈들에 대한 인덱스는 다음과 같은 방법으로 계산된다.

$$MMS(x,y,z,type)=\mu(x,y,z)+\delta(type,k),\ 0\le k<n$$

그리고, 17가지 접근형태($type$)에 따른 $\delta(k)$는 다음과 같다.

$XB,\ XYS,\ ZXS,\ Cube$: $\delta(k)=k//m,\ 0\le k<n,$

$YB,\ YZS$: $\delta(k)=pk//m,\ 0\le k<n,$

ZB : $\delta(k)=pqk//m,\ 0\le k<n,$

$XYFD$: $\delta(k)=((p+1)k)//m,\ 0\le k<n,$

$YZFD$: $\delta(k)=((q+1)k)//m,\ 0\le k<n,$

$ZXFD$: $\delta(k)=((pq+1)k)//m,\ 0\le k<n,$

$XYBD$: $\delta(k)=((p-1)k)//m,\ 0\le k<n,$

$YZBD$: $\delta(k)=((q-1)k)//m,\ 0\le k<n,$

$ZXBD$: $\delta(k)=((pq-1)k)//m,\ 0\le k<n,$

$XYZMjD$: $\delta(k)=((pq+p+1)k)//m,\ 0\le k<n,$

$XZMjD$: $\delta(k)=((pq-p+1)k)//m,\ 0\le k<n,$

$YZMjD$: $\delta(k)=((pq+p-1)k)//m,\ 0\le k<n,$

$XYMjD$: $\delta(k)=((pq-p-1)k)//m,\ 0\le k<n.$

4.1절의 예를 토대로 기저좌표 (0,0,0)과 접근방식이 YB인 경우, n=8이기 때문에 논리적인 좌표들 (0,0,0), (0,1,0), (0,2,0), (0,3,0), (0,4,0), (0,5,0), (0,6,0), (0,7,0)에 저장된 각각의 데이타는 물리적인 메모리 모듈 0, 2, 4, 6, 8, 10, 1, 3 에 각각 저장된다. 그래서 이들 데이타에 동시접근하기 위해서 메모리 모듈 선택에 의해 계산된 메모리 모듈들을 활성화시켜야 한다.

4.4 데이타 라우팅

n개의 외부 데이타 레지스터($D1$)에 저장된 데이타를 m개의 메모리 모듈들 중 해당되는 n개의 메모리 모듈들에 저장하기 위해서는 접근방식별로 재정렬을 위한 라우팅 과정을 거친다. 라우팅된 n개의 데이타는 m개의 메모리 모듈들에 직접 연결된 임시 레지스터($D2$)에 저장된다. 여기서 $D1$레지스터의 크기는 n이며 $D2$레지스터의 크기는 m이 된다. 17가지 접근방식별 라우팅을 위한 함수들은 다음과 같다.

$XB,\ XYS,\ ZXS,\ Cube$: $D2(k//m)\leftarrow D1(k),\ 0\le k<n,$

$YB,\ YZ$: $D2((pk)//m)\leftarrow D1(k),\ 0\le k<n,$

ZB : $D2((pqk)//m)\leftarrow D1(k),\ 0\le k<n,$

$XYFD$: $D2(((p+1)k)//m)\leftarrow D1(k),\ 0\le k<n,$

$YZFD$: $D2(((q+1)k)//m)\leftarrow D1(k),\ 0\le k<n,$

$ZXFD$: $D2(((pq+1)k)//m)\leftarrow D1(k),\ 0\le k<n,$

$XYBD$: $D2(((p-1)k)//m)\leftarrow D1(k),\ 0\le k<n,$

$YZBD$: $D2(((q-1)k)//m)\leftarrow D1(k),\ 0\le k<n,$

$ZXBD$: $D2(((pq-1)k)//m)\leftarrow D1(k),\ 0\le k<n,$

$XYZMjD$: $D2(((pq+p+1)k)//m)\leftarrow D1(k),\ 0\le k<n,$

$XZMjD$: $D2(((pq-p+1)k)//m)\leftarrow D1(k),\ 0\le k<n,$

$YZMjD$: $D2(((pq+p-1)k)//m)\leftarrow D1(k),\ 0\le k<n,$

$XYMjD$: $D2(((pq-p-1)k)//m)\leftarrow D1(k),\ 0\le k<n.$

상기 함수들은 외부 입력 데이타를 메모리 시스템에 저장하기 위한 쓰기($WRITE$) 메모리 연산일 경우이며, 반대로 읽기($READ$) 메모리 연산일 경우는 상기 함수의 역함수를 사용한다.

4.4절의 예를 토대로 쓰기 메모리 연산일 경우, 논리적인 좌표들에 해당하는 8개의 데이타가 $D1(0)$에서 라

우팅 되어 $D2(0)$에, $D1(1)\rightarrow D2(2)$, $D1(2)\rightarrow D2(4)$, $D1(3)\rightarrow D2(6)$, $D1(4)\rightarrow D2(8)$, $D1(5)\rightarrow D2(10)$, $D1(6)\rightarrow D2(1)$, $D1(7)\rightarrow D2(3)$에 각각 저장된 후 메모리 모듈 0, 1, 2, 3, 4, 6, 8, 10에 각각 저장된다.

4.5 주소 계산 및 라우팅

m개의 메모리 모듈들 중 서로 다른 n개의 메모리 모듈들에 저장된 n개의 데이타에 동시접근하기 위해서는 n개의 메모리 모듈들을 활성화시킴과 동시에 동시접근하려는 n개의 데이타에 해당하는 주소들을 계산해야 한다. 4.3절에서 언급한 바와 같이 데이타에 접근하기 위해서 외부에서 기저좌표(x,y,z)와 접근방식$(type)$가 지정되어야 하며 17가지 접근방식에 따른 주소계산은 다음과 같다.

$$XBA(x,y,z)=(y/(qr))\cdot s1+(x+k)/p+z\cdot s2,$$

$$YBA(x,y,z)=((y+k)/(qr))\cdot s1+x/p+z\cdot s2,$$

$$YBA(x,y,z)=(y/(qr))\cdot s1+x/p+(z+k)\cdot s2,$$

$$XYFDA(x,y,z)=((y+k)/(qr))\cdot s1+(x+k)/p+z\cdot s2,$$

$$YZFD(x,y,z)=((y+k)/(qr))\cdot s1+x/p+(z+k)\cdot s2,$$

$$ZXFDA(x,y,z)=(y/(qr))\cdot s1+(x+k)/p+(z+k)\cdot s2,$$

$$XYBDA(x,y,z)=((y+k)/(qr))\cdot s1+(x-k)/p+z\cdot s2,$$

$$YZBDA(x,y,z)=((y-k)/(qr))\cdot s1+x/p+(z+k)\cdot s2,$$

$$ZXBDA(x,y,z)=(y/(qr))\cdot s1+(x-k)/p+(z+k)\cdot s2,$$

$$XYZMjDA(x,y,z)=((y+k)/(qr))\cdot s1+(x+k)/p+(z+k)\cdot s2,$$

$$XZMjDA(x,y,z)=((y-k)/(qr))\cdot s1+(x+k)/p+(z+k)\cdot s2,$$

$$YZMjDA(x,y,z)=((y+k)/(qr))\cdot s1+(x-k)/p+(z+k)\cdot s2,$$

$$XYMjDA(x,y,z)=((y-k)/(qr))\cdot s1+(x-k)/p+(z+k)\cdot s2,$$

$$XYSA(x,y,z)=((y+k/p)/(qr))\cdot s1+(x+k//p)/p+z\cdot s2,$$

$$YZSA(x,y,z)=((y+k//q)/(qr))\cdot s1+x/p+(z+k/q)\cdot s2,$$

$$ZXSA(x,y,z)=(y/(qr))\cdot s1+(x+k//(pq))/p+(z+k/(pq))\cdot s2,$$

$$\begin{aligned}CubeA(x,y,z)=&((y+(k//(pq))/p)/(qr))\cdot s1\\&+(x+(k//(pq))//p)/p2\\&+(z+k/(pq))\cdot s2,\\&0\le k<pqr.\end{aligned}$$

상기 함수들에 의해 계산된 n개의 주소들을 데이타 라우팅과 동일한 방법으로 라우팅함으로써 n개의 서로 다른 메모리 모듈에 저장된 해당 데이타에 대한 동시접근을 허용하게 된다.

앞서 언급된 예를 통해서 논리적인 좌표들 (0,0,0), (0,1,0), (0,2,0), (0,3,0), (0,4,0), (0,5,0), (0,6,0), (0,7,0)에 저장된 각각의 데이타는 물리적인 메모리 모듈 0, 2, 4, 6, 8, 10, 1, 3 의 주소 0, 0, 0, 0, 4, 4, 4, 4 에 각각 저장된다.

4.6 간격을 허용하는 메모리 시스템

본 논문에서 제안하는 메모리 시스템은 지정된 임의의 좌표에서 간격(t)를 갖고 17가지 접근방식 내의 데이타에 동시접근 할 수 있다. 이 때 간격 t는 $t>0$이고 $t(\mathrm{mod}m)\neq 0$인 조건을 만족하는 임의의 자연수이다. 간격 t를 허용하는 메모리 시스템은 4.3절 메모리 모듈 선택, 4.4 데이타 라우팅, 4.5 주소 계산 및 라우팅에서 기술된 수식들에서 변수 k를 $k\times t$로 대체함으로써 구해질 수 있다.

4.7 메모리 시스템 설계

제안하는 메모리 시스템을 설계할 경우, 응용 분야에서 요구하는 성능을 얻기 위해서 앞서 언급했던 디자인 파라메터를 고려해야 한다. 앞서 언급했던 바와 같이 초기에 고려해야 하는 디자인 파라메터들로는 L,M,N,p,q,r,m과 접근방식 및 접근간격 등이 있으며, 추가적으로,

1) $\lceil \log_2(L/p+(LM+LMN)/pqr)\rceil \times d-bit$ 메모리 모듈,
2) {접근방식과 접근간격}을 고려한 t-bit $type$-레지스터,
3) $\lceil \log_2 L\rceil$-bit x-레지스터, $\lceil \log_2 M\rceil$-bit y-레지스터, $\lceil \log_2 N\rceil$-bit z-레지스터,
4) 외부장치와의 데이타 전송을 위한 pqr개의 d-bit 레지스터,
5) 외부 메모리 모듈과의 데이타 전송을 위한 m개의 d-bit 레지스터,
6) m개의 메모리 모듈들에서 pqr개의 메모리 모듈들을 활성화(enable)시키는 회로,
7) 각각의 활성화된 메모리 모듈의 주소를 계산하는 회로,
8) 외부로부터 전송된 데이타를 활성화된 메모리 모듈로 라우팅하는 회로,
9) 외부 인터페이스 및 메모리 모듈 제어를 위한 회로 등이 고려되어야 한다.

이들을 고려하여 본 논문에서 제안하는 3차원 메모리 시스템은 메모리 모듈 선택회로, 데이타 라우팅 회로, 주소 계산 및 라우팅 회로, 그리고 외부 장치들과의 상호작용을 위한 인터페이스 등으로 구성될 수 있으며, 이들로 구성된 메모리 시스템의 블록도는 그림 5와 같다.

4.3절에 언급된 접근방식별 메모리 모듈 선택을 위한 계산식들은 모듈러 연산을 수행하기에 구현 시 회로가 복잡해진다. 그 결과 시간 복잡도가 높아지기 때문에 그림 5의 메모리 모듈 선택 회로에 $\lceil \log_2(L+M/p+N/pq)\rceil \times (\log_2 M\times t)$-bit ROM 또는 Latch을 두어 미리 계산된 결과값을 저장함으로써 이러한 문제를 해결할 수 있다. 주소

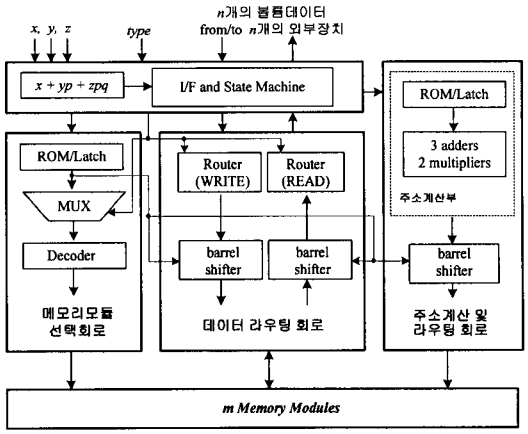

그림 5 본 논문에서 제안하는 3차원 메모리 시스템의 블록도

계산부 역시 4.5절에서 언급된 것과 같이 복잡한 수식 연산을 수행하기 때문에 구현에 있어 시간복잡도를 고려해야 한다. 그래서 그림 5의 주소 계산 및 라우팅 회로의 주소 계산부에 최대 $t\times\lceil\log_2(L/p+(LM+LMN)/pqr)\rceil\times m$ -bit ROM 또는 Latch를 두어 접근방식별 기저주소를 저장한다. 접근방식별 기저주소는 기저좌표(x,y,z)를 기준으로 연속된 pqr개의 데이타에 해당하는 주소들 중 가장 작은 주소값들의 배열을 선택함으로써 구해질 수 있다[14]. 이렇게 함으로써 복잡한 수식 연산을 하나의 시스템 클럭에 수행할 수 있다. 그리고 데이타 라우팅을 위해서는 $m\times d$개 만큼의 쉬프트를 위한 회로가 필요한데, 이 때 쉬프트 레지스터 대신 배럴 쉬프터를 사용함으로써 쉬프트 연산을 하나의 시스템 클럭에 수행할 수 있다. 만약, 제안된 3차원 메모리 시스템 설계 시 전체 성능을 위해 고려해야 하는 파라메터들인 L,M,N,p,q,r이 모두 2의 제곱이라면 모듈선택 및 주소계산을 위한 연산 회로는 간단해진다. 왜냐하면 곱셈 및 나눗셈과 모듈러 연산들은 쉬프트 연산 또는 직접적인 선(wire) 연결만으로 해결될 수 있기 때문이다.

제안하는 3차원 메모리 시스템을 설계하는데 있어 추가적으로 고려해야 할 부분은 외부 인터페이스 및 메모리 모듈을 제어하는 부분이다. 메모리 시스템 자체가 논리적인 메모리 시스템이기 때문에 외부 인터페이스는 일반적인 메모리 인터페이스로 구성되면 되지만, 메모리 모듈을 제어하는 부분은 어떤 특성의 메모리를 사용하는냐에 따라 달라지게 된다. 즉, 메모리 모듈을 SRAM 또는 DRAM으로 사용하느냐에 따라 메모리 제어 부분의 설계가 달라지게 된다. 본 논문에서는 메모리 모듈을 SRAM으로 간주하여 상태기(state machine)로 구성된 메모리 제어부를 설계하고 일반적인 SRAM 제어를 위한 cs_, oe_, we_, re_ 신호들을 출력하게 하였다.

4.8 성능 분석

본 논문에서 제안하는 메모리 시스템은 Verilog-HDL로 설계하였으며 기능 검증은 하드웨어 모의실험 패키지인 CADENCE사의 Verilog-XL로 기능단계 모의실험(Functional Level Simulation)을 수행하였다.

기존의 메모리 시스템들[9-13]과 본 논문에서 제안하는 메모리 시스템의 성능 및 기능 비교는 초당 처리 샘플수를 나타내는 SPPS(Sample Processed Per Second)[6]와 제안하는 메모리 시스템에서 지원하는 17가지 접근방식에 대한 최고(Max), 평균(Mean), 최저(Min)치의 메모리 접근 주기를 대상으로 했으며 비교 결과는 표 1과 같다.

SPPS는 데이타 공간이 256^3이고 시스템 클럭이 66MHz일 경우를 기준으로 제안된 메모리 시스템의 디자인 파라메터 L,M,N,p,q,r를 각각 256, 256, 256, 2, 2, 2로 설정하였다. 이는 [6]에서 언급된 방법과 동일한 조건으로 제안된 메모리 시스템의 성능을 비교 분석하기 위한 것으로써 pqr=8, 즉 메모리 모듈의 개수를 8로 하였으며 외부 프로세서의 개수도 8개로 가정하였다. 이 때 병렬로 8개의 외부 프로세서가 처리할 수 있는 샘플의 개수는 초당 $(8\times66\times10^6)/4$ sample/sec 이다. 그러나 제안된 메모리 시스템은 READ 명령을 수행할 경우 4-stage pipelined machine으로 동작하기 때문에 264×10^6 sample/sec 만큼 수행할 수 있다.

기존의 메모리 시스템들은 볼륨렌더링 시스템을 위한

표 1 기존의 메모리 시스템과 제안된 메모리 시스템의 비교

	VIZARD II [9]	VOGUS [10]	VIRIM [11]	Cube-4 [16]	EM-Cube[13]	The Proposed
Memory Partitioning	4-way interleaved	8-way interleaved	8-way interleaved	Skewed	Skewed Block	Skewed
Published SPPS[10]	56×10^6 (m=4)	40×10^6 (m=8)	40×10^6 (m=8)	528×10^6 (m=16)	533×10^6 (m=4)	264×10^6 (m=8)
Max	8	4	4	4	4	1
Mean	3.47	3.82	3.82	1.35	3.65	1
Min	1	1	1	1	1	1

전용 메모리 시스템으로 설계되었기 때문에 적용된 알고리즘들에 따라 몇 가지 특정 접근방식에 대한 메모리 접근 주기가 1에 해당한다. 하지만 어느 정도의 범용성을 고려할 경우, 다양한 접근 방식을 요구하는 여러 응용분야에 적용한다면 이들 기존의 메모리 시스템들은 본 논문에서 제안하는 메모리 시스템에 비해 충분한 성능을 발휘하지 못한다. 이는 특정 접근형태에 대한 메모리 접근 주기의 개수를 비교함으로써 확인할 수 있으며 결과는 표 1과 같다. 본 논문에서 제안하는 17가지 접근방식은 기존의 메모리 시스템들이 제공하는 모든 접근방식을 포함하고 있으며 17가지 접근방식 모두 동일한 메모리 접근 주기 1에 해당하는 시간을 요구한다.

5. 결 론

본 논문에서는 다양한 응용 분야에서 널리 사용될 수 있도록 17가지 접근방식을 지원하는 3차원 메모리 시스템을 제안하였다. 제안한 메모리 시스템은 기본 함수인 메모리 모듈 할당 함수와 주소 할당 함수를 토대로 메모리 모듈 선택 회로, 데이타 라우팅 회로, 주소 계산 및 라우팅 회로 등으로 구성되며 그림 5에서 시스템 블록도를 제시하였다. 또한 메모리 모듈 할당 및 주소 계산을 위한 수식들을 회로로 구현함에 있어 발생될 수 있는 복잡성을 고려한 대체 방안도 제시하였다.

본 논문에서 언급된 기존의 메모리 시스템들은 볼륨 렌더링을 실시간(256^3 data space, 30 frame/sec)으로 수행하기 위한 전용 시스템의 메모리 시스템으로 연구 개발되었으며 이들은 블록 메모리와 벡터 메모리 구조로 되어있다. 본 논문에서 제안하는 메모리 시스템은 단일 접근방식만을 지원하는 블록 메모리의 단점과 사각형 및 육면체 접근방식을 지원하지 못하기 때문에 다양한 응용 분야를 위한 시스템 구성에 있어 한계성을 내포한 벡터 메모리의 단점을 모두 해결한 것으로써, 볼륨 렌더링 이외의 다양한 응용 분야에 활용될 수 있다. 또한 제안한 메모리 시스템은 임의의 간격을 허용하기 때문에 다중해상도를 지원하는 프레임 버퍼로써도 활용될 수 있다.

참 고 문 헌

[1] D. C. Van Voorhis and T. H. Morrin, "Memory Systems for Image Processing," *IEEE Transactions on Computers*, vol. C-27, no. 2, pp.113-125, Feb. 1978.

[2] D. H. Lawrie and C. R. Vora, "The Prime Memory System for Array Access," *IEEE Transactions on Computers*, vol. C-31, no. 5, pp.435-442, May 1982.

[3] J. W. Park, "Mutliaccess Memory System for Attached SIMD Processor," *IEEE Transactions on Computers*, vol. 53, no. 3, pp.1-14, Mar. 2004.

[4] A. Kaufman, "Volume Visualization," *IEEE CS Press Tutorial*, Los Alamitos, Calif., 1991.

[5] D. Cohen and A. Kaufman, "A 3D Skewing and De-skewing Scheme for Conflict-Free Access to Rays in Volume Rendering," *IEEE Transactions on Computers*, vol. 44, no. 5, pp. 707-710, May 1995.

[6] H. Ray, H. Pfister, D. Silver, and T. A. Cook, "Ray Casting Architecture for Volume Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 3, pp. 210-223, July. 1999.

[7] M. de Boer, A. Gropl, J. Hesser, and R. Maenner, "Latency- and Hazard-Free Volume Memory Architecture for Direct Volume Rendering," *Computer and Graphics*, vol. 21, no. 2, pp. 179-187, 1997.

[8] M. Doggett and M. Meissner, "A Memory Addressing and Access Design for Real Time Volume Rendering," *Proc. 1999 IEEE International Symppsium on Circuit and Systems*, vol 4, pp. 344-347, 1999.

[9] M. Meissner, U. Kanus, and W. Strasser, "VIZARD II : A PCI-Card for Real-Time Volume Rendering," Proc. SIGGRAPH/Eurographics Workshop Graphics Hardware, pp. 61-67, Aug. 1998.

[10] G. Knittel, "A Scalable Architecture for Volume Rendering," *Computer and Graphics*, vol. 19, no. 5, pp. 653-665, 1995.

[11] T. Guenther, C. Poliwoda, C. Reinhard, J. Hesser, R. Maenner, H. P. Meinzer, and H. J. Baur, "VIRIM : A Massively Parallel Processor for Real-Time Volume Visualization in Medicine," *Proc. the 9th Eurographics Hardware Workshop*, vol. 19, no. 5, pp. 705-710, 1995.

[12] H. Pfister and A. Kaufman, "Cube-4 - A Scalable Architecture for Real-Time Volume Rendering," *Proc. Volume Visualization Symp.*, pp. 47-54, Oct. 1996.

[13] R. Osborne, H. Pfister, H. Lauer, N. McKenzie, S. Gibson, W. Hiatt, and T. Ohkami, "EM-Cube: An architecture for low-cost real-time volume rendering," *Proc. SIGGRAPH/Eurographics Workshop Graphics Hardware*, pp. 131-138, Log Angeles, Calif., Aug. 1997.

[14] 이형, A 3D Memory System for Conflict-Free Access, 충남대학교 박사학위논문, Feb. 2005.

이 형

1995년 충남대학교 컴퓨터과학과 이학사 1997년 충남대학교 컴퓨터공학과 공학석사. 2005년 충남대학교 컴퓨터공학과 공학박사. 2001년~현재 대전보건대학 방송제작기술과 전임강사. 관심분야는 비디오 처리, 병렬처리