http://dx.doi.org/10.5392/JKCA.2012.12.01.030

처리기에 지역 버퍼 메모리 시스템을 지원하는 다중접근기억장치

Multiaccess Memory System supporting Local Buffer Memory System to Processing Elements

이 형
대전보건대학교 방송제작과

Hyung Lee(hyung@hit.ac.kr)

요약

선형 비틀림 구조를 갖는 메모리 시스템은 SIMD 구조에 적합한 메모리 시스템으로써, 2차원 데이터 배열인 $M \times N$에서 임의의 위치로부터 임의의 간격을 갖고 다양한 접근형태들로, m개의 메모리 모듈들에서 n개의 데이터를 동시에 접근할 수 있다. 그러나 이러한 메모리 시스템은 논리적인 2차원 $M \times N$ 데이터 배열을 지원하기 위해 $m \times cells$의 물리적인 메모리 용량이 필요하지만, 적어도 $(m-n) \times cells$ 만큼의 메모리 셀은 사용되지 않는다. 여기서 $cells$는 $(M-1)/q+(N-1)/p \times \lceil M/q \rceil +1$이다. 본 논문에서는 이러한 메모리 시스템의 모든 기능들을 수용하면서 $t>0$인 조건 하에 사용되지 않는 메모리 셀들 중 $(n \times t) \times N/p$ 만큼을 n개의 처리기들에 지역 버퍼 메모리로 제공할 수 있는 방법을 제안한다

■ 중심어 : | 병렬메모리 | 다중접근기억장치 | SIMD 구조 |

Abstract

A memory system with the linear skewing scheme has been regarded as one of suitable memory systems for a single instruction, multiple data (SIMD) architecture. The memory system supports simultaneous access n data to m memory modules within various access types with a constant interval in an arbitrary position in two dimensional data array of $M \times N$. Although $m \times cells$ memory cells are physically required to support logical two dimensional $M \times N$ array of data by means of the memory system, at least $(m-n) \times cells$ memory cells remain in disuse, where $cells$ is $(M-1)/q +(N-1)/p \times \lceil M/q \rceil +1$. On keeping functionalities the memory system supports, $(n \times t) \times N/p$ out of a number of unused memory cells, where $t>0$, being used as local buffer memories for n processing elements is proposed in this paper

■ keyword : | Parallel Memory | Multiaccess Memory System | SIMD Architecture |

I. 서 론

방대한 양의 데이터를 실시간으로 처리하기 위한 고성능 컴퓨터에 대한 연구가 프로세서와 메모리 시스템, 그리고 프로그래밍 관점 등에서 활발히 진행되고 있다. 프로세서의 경우, 중앙처리장치는 다중코어 (multi core)로, 그래픽처리장치는 다수의 코어 (many-core)로 집중되면서 특정 분야에서 한정되어 연구되던 병렬처리구조, 특히 SIMD (Single Instruction, Multiple Data) 구조에 대한 관심이 재차 대두되고 있다. 다수의

접수번호 : #110809-010
접수일자 : 2011년 08월 09일
심사완료일 : 2011년 11월 11일
교신저자 : 이형, e-mail : hyung@hit.ac.kr

처리기들로 구성된 컴퓨터의 경우, 처리기들이 처리할 방대한 양의 데이터를 지원하기 위해 메모리 대역폭을 확보하는 병렬 메모리 시스템에 대한 연구들이 병행되고 있다. 또한 이러한 구조가 지원할 수 있는 최고의 성능을 확보하기 위해 프로그래밍 모델 관점에서도 병렬처리기법에 대한 다양한 연구가 진행되고 있다[1][2].

고성능 컴퓨팅을 위한 메모리 대역폭을 확보하기 위한 메모리 시스템들은 일반적으로 다수의 메모리 모듈들을 구성하여 처리기들이 동시에 접근하는 것이다. 다수의 메모리 모듈들로 메모리 시스템을 구축하는 방법 중의 하나로 메모리 모듈들에 대한 연속적인 메모리 요청을 중첩시키는 것이다[3]. 하지만 이 방법은 다수의 메모리 모듈들에 동시에 접근할 수 없기 때문에 SIMD 구조에 적합하지 않다. SIMD 구조를 위한 단순한 메모리 시스템은 접근하려는 데이터의 주소(a)를 메모리 모듈의 주소($a\%m$)에 매핑시키는 것이다. 여기서 m은 메모리 모듈들의 개수로서 처리기들의 개수(n)와 일치할 때 최상의 성능을 발휘할 수 있으며, 구현 상의 복잡도를 고려하여 2의 승수를 취한다. 그러나 이러한 인터리브드 방식의 메모리 시스템들[4][5]은 일정한 간격과 다수의 접근형태들이 요구되는 상황에서는 동시에 다수의 메모리 모듈들에 접근 시 충돌 문제 등이 발생하며 이를 해결하기 위한 추가된 기능 등으로 낮은 성능을 보였다. 이를 해결하기 위해 비선형 비틀림 (non-linear skewing) 메모리 시스템[6][7]에 대한 연구가 진행 되었지만 이들 역시 한정된 접근형태들과 간격들, 그리고 접근 위치를 제한시켰다. 초기의 선형 비틀림 (linear skewing) 메모리 시스템[8]이 제안되면서 많은 연구자들이 이를 토대로 다양한 접근형태들과 임의의 간격을 지원하기 위한 연구들이 진행되었다[9-12]. 최근에는 앞서 언급했던 메모리 구조들의 장단점들을 고려하여, 특정 응용에 적합한 컴퓨터 시스템들을 개발하면서 그 응용에 최적화된 메모리 시스템에 대한 연구들이 진행되고 있다. 예를 들면, 특정 접근형태에 임의의 간격들만을 허용하고[13], 충돌 발생이 감지되면 직렬 접근을 수행하며[14], 파이프라인 기법을 접목하여 계산된 주소를 큐로 관리하여 충돌을 회피하는[17] 방법들 등이 있다.

일반적으로 다양한 접근형태들과 임의의 간격들을 지원하기 위해 메모리 모듈들의 개수(m)와 동시에 접근하는 데이터의 개수(n)가 $m \geq n$인 조건에서 충돌없이 동시에 n개의 데이터에 접근할 수 있다. 이를 위해서는 기본적으로 메모리모듈할당함수 $\mu(\cdot)$와 주소계산 및 할당함수 $\alpha(\cdot)$가 고려되어야 한다. 논리적인 2차원 데이터 배열인 $M \times N$ 내의 임의의 좌표인 $I(x, y)$는 이들 함수들을 토대로 $f: I(x, y) \Rightarrow (\mu(\cdot), \alpha(\cdot))$에 의해 특정 메모리 모듈의 특정 셀에 매핑되며, 특히 $\alpha(\cdot)$는 다음의 관계를 만족시켜야 한다.

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

다양한 접근형태들과 임의의 간격을 지원하는 메모리 시스템들의 단점은 주소를 계산하는 회로가 복잡하다는 것이다. 이에 빠른 주소 계산 방법과 좀 더 다양한 접근형태들 및 기존의 제한적인 요건들을 완화시킨 다중접근기억장치가 연구되었다[12][16]. 또한 이렇게 연구된 다중접근기억장치는 방대한 양의 데이터를 처리하기 위한 SIMD 구조에 활용되어 다양한 응용에 적용되었고[17]과 시제품으로 제작되어 그 성능을 입증하였다[18].

다중접근기억장치[16-18]는 논리적으로 2차원 데이터 배열인 $M \times N$ 내의 임의의 좌표를 기준으로 임의의 간격을 허용하면서 접근형태 내의 n개의 데이터를 물리적으로 구성된 m개의 메모리 모듈들에서 동시에 접근하는 기능을 제공한다. 또한 특정 접근형태와 특정 간격으로 제한하지 않고 다양한 접근형태들과 임의의 간격을 동시에 지원하기 위해서 다중접근기억장치는 m을 n에 근접한 솟수로 설정하여 메모리 모듈들의 개수를 최소화하였다. 하지만 논리적으로 지원하는 2차원 데이터 배열인 $M \times N$에 해당하는 실제적인 메모리 셀들의 개수는 $n \times (\alpha(M-1,\ N-1)+1)$이지만 다양한 기능들을 제공하기 위해서는 $m \times (\alpha(M-1,\ N-1)+1)$인 메모리 셀들을 요구한다. 그래서 동일 메모리 주소에 대해 적어도 $m-n$개의 메모리 셀들이 사용되지 않는다.

이에 본 논문에서는 이들 사용되지 않는 메모리 셀들을 활용하는 방법을 제안하며, 제 2장에서 기존의 다중

접근기억장치를 기술하고 제 3장에서는 추가적인 메모리 활용방법을 제안한다.

II. 다중접근기억장치

2차원 데이터 배열인 $M \times N$ 내의 임의의 좌표인 $I(x, y)$를 기준으로 임의의 간격(r)을 갖고 블록, 행, 열, 대각선, 역대각선 등의 다양한 접근형태들 내의 n개의 데이터를 m개의 메모리 모듈들에서 동시에 접근할 수 있는 다중접근기억장치[16]의 구성은 다음과 같다.

1. 기본 함수들

기저좌표인 $I(x, y)$에 해당하는 메모리 모듈을 결정하는 함수는 수식(1)이며, 그에 대응하는 주소를 계산하고 할당하는 함수는 수식(2)이다.

$$\mu(x,y) = (x+qy)\%m,\ 0 \leq x < M,\ 0 \leq y < N. \quad (1)$$

$$\alpha(x,y) = x/p + (y/q) \cdot s,\ 0 \leq x < M,\ 0 \leq y < N. \quad (2)$$

여기서 p와 q는 설계 인자이며, 동시에 접근하는 데이터의 개수 n은 $p \times q$이다. m은 메모리 모듈들의 개수이며 n보다 크면서 n에 가장 근접한 소수이다. 그리고 s는 $\lceil M/q \rceil$ 인 음이 아닌 정수이다. 참고로, 본 논문에서 기술되는 모든 변수들(피연산자)와 모든 연산의 결과는 음수가 아닌 정수이다.

2. 임의의 간격 r을 갖는 다양한 접근형태들

다중접근기억장치가 지원하는 접근형태들은 블록(*BLK*), 행(*ROW*), 열(*COL*), 대각선(*FRD*), 역대각선(*BKD*)이며, $r > 0$와 $r\%m \neq 0$인 조건을 만족하는 임의의 간격(r)를 갖고 다양한 접근형태들 내의 n개의 데이터들을 m개의 메모리 모듈들에서 동시에 접근하기 위해 n개의 메모리 모듈들을 선택하고, 접근할 n개의 데이터에 대한 주소들을 계산하는 함수들은 다음과 같다.

$$BLK(x,y,r):\ \mu(x+ra,\ y+rb),\ \alpha(x+ra,\ y+rb),$$
$$ROW(x,y,r):\ \mu(x+rk,\ y),\ \alpha(x+rk,\ y),$$
$$COL(x,y,r):\ \mu(x,\ y+rk),\ \alpha(x,\ y+rk),$$
$$FRD(x,y,r):\ \mu(x+rk,\ y+rk),\ \alpha(x+rk,\ y+rk),$$
$$BKD(x,y,r):\ \mu(x-rk,y+rk),\ \alpha(x-rk,\ y+rk).$$

여기서 변수 a와 b의 범위는 각각 $0 \leq a < p$, $0 \leq b < q$이고, 변수 k의 범위는 $0 \leq k < n$이다. 사용되는 변수의 일관성을 고려하여 $BLK(x,y,r)$는 k에 대한 메모리모듈할당과 주소할당 함수들은 $\mu(x+(k\%q)r,\ y+(k/q)r)$와 $\alpha(x+(k\%q)r,\ y+(k/q)r)$로 각각 대체시킬 수 있다.

3. 데이터와 주소 라우팅

다중접근기억장치에서 쓰기 연산은 m개의 메모리 모듈에서 n개를 선택하여 n개의 외부 데이터를 저장하는 것이다. 이를 위해서는 저장할 데이터와 주소할당함수에 의해서 계산된 주소들을 각각 재정렬시켜 메모리 모듈할당함수에 의해 결정된 n개의 메모리 모듈들에 매핑시켜야 한다. 이를 위한 접근형태별 라우팅 함수들은 다음과 같으며, 이들 함수들로 데이터와 계산된 주소들을 각각 재정렬시킨 후 $\mu(x,y)$만큼 right-shift를 수행한다.

$$BLK, ROW:\ RO(rk\%m) \leftarrow RI(k),$$
$$COL:\ RO(rq\%m) \leftarrow RI(k),$$
$$FRD:\ RO(((q+1)rk)\%m) \leftarrow RI(k),$$
$$BKD:\ RO(((q-1)rk)\%m) \leftarrow RI(k).$$

여기서 *RI*는 외부와 연결된 임시 저장소로 크기는 n이고, *RO*는 메모리 모듈들과 연결된 임시 저장소로 크기는 m이며, 변수 k의 범위는 $0 \leq k < n$이다.

2차원 배열이 $M \times N = 20 \times 20$인 경우에, $M \times N$은 [그림 1]-(a)와 같은 데이터를 갖는 배열로 구성될 수 있으며, 디자인 인자들이 $p = q = 2$이고 $m = 5$인 다중접근기억장치를 구현했을 때, 동시에 접근할 수 있는 데이터의 개수는 $n(=4=p \times q)$이다. 이 때 $STORE(x, y, r, type)$ 명령어 형태로 다중접근기억장치에 데이터를 저장할 수 있다. 예를 들면, $x = 1,\ y = 0,\ r = 1, type = BLK$ 는 기저좌표 $I(1,0)$에 간격이 1인 블

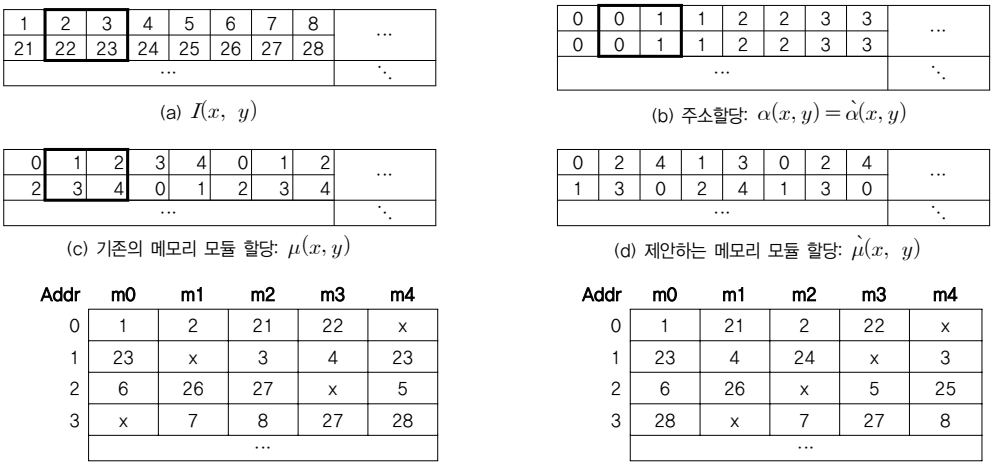


그림 1. $M\times N=20\times 20, p=q=2,\ m=5$**인 경우,** $I(x,\ y)$**에 대한 주소할당 및 수식(1)과 수식(3)에 따른 메모리 모듈 할당과 이에 따른 메모리 모듈 내의 데이터 구성**

록접근형태 내의 2, 3, 22, 23인 데이터를 동시에 접근하게 된다([그림 1]-(a)의 굵은 사각형). 접근형태가 블록이기에 선택된 메모리모듈들의 번호는 1, 2, 3, 4이고([그림 1]-(c)), 계산된 주소는 0, 1, 0, 1 이다([그림 1]-(b)). 즉, $I(1,0)$에 해당하는 데이터 2는 메모리 모듈 1번의 주소 0에 저장된다. 데이터 및 주소 라우팅은 블록접근형태의 라우팅함수에 의해 메모리모듈 1, 2, 3, 4가 1, 2, 3, 4, x로, 주소 0, 1, 0, 1이 0, 1, 0, 1, x로 라우팅된 후 $\mu(1,0)=1$이기 때문에 1만큼 right-shift하여 메모리 모듈들은 x, 1, 2, 3, 4로, 주소들은 x, 0, 1, 0, 1로 재배열 된다. 이 때 x는 don't care이다. 결과적으로, 블록접근형태 내의 데이터 2, 3, 22, 23은 메모리모듈 1 ,2 ,3 ,4의 주소 0, 1, 0, 1에 각각 저장된다. 읽기 연산은 상기 과정들을 역순으로 수행한다.

III. 제안하는 다중접근기억장치

특정한 용도의 응용이나 일반적인 병렬 프로그램을 위해 다중접근기억장치를 설계할 경우, 처리할 데이터를 저장하기 위해서는 $M\times N$의 크기와 처리할 데이터의 개수는 n이기 때문에 M, N, p, q 인자들이 우선적인 고려의 대상이 되어야 한다. 일반적으로 고려해 볼 때 n의 크기가 클수록 시간당 데이터 처리율을 높일 수 있지만 메모리 모듈들의 개수인 m도 그만큼 커지게 된다. 또한 n를 고정시키고 $M\times N$의 크기를 증가시키면 메모리 모듈당 용량이 증가하게 된다. 이러한 경우들은 구현 시 데이터 버스와 주소 버스 및 제어 신호들의 개수와 밀접한 관계가 있기 때문에 메모리 모듈들을 포함하는 다중접근기억장치는 하나의 칩에 집적되는 것이 효율적이다. 하지만 하나의 칩에 집적되기에는 $M\times N$의 크기에 제한이 있으나 향 후 기술력이 향상되어 칩의 집적도가 높아진다면 많은 내부 메모리 용량을 지원하여 더 큰 용량의 $M\times N$을 구성할 수 있다.

다중접근기억장치는 각각의 메모리 모듈당 $\alpha(0,0)$부터 $\alpha(M-1,\ N-1)$까지의 주소를 지원한다. 즉, 논리적인 $M\times N$ 데이터 배열을 제공하기 위해 실제적으로 구성되는 메모리의 용량은 $(\alpha(M-1,\ N-1)+1)\times m$이며 이 수치는 $M\times N$보다 항상 크다. 이는 동일한 주소에 대해 m개의 메모리 모듈들 중에서 n개만 사용하기 때문에 적어도 $(m-n)\times(\alpha(M-1,\ N-1)+1)$의 메모리 셀들은 사용되지 않는다. 그렇기 때문에 $m-n$가 작을수록 메모리의 활용도가 높고, 추가적으로 주소할당함수가 $p\times q$ 배열 내에서는 서로 다른 메모리모듈에 대해

서 동일한 주소를 갖기 때문에 $N\%p=0$이고 $M\%q=0$일 때 메모리의 활용도가 높다.

본 논문에서는 메모리 모듈들을 포함한 다중접근기억장치를 하나의 칩에 집적시킬 경우 제 2장에서 언급한 다중접근기억장치의 모든 기능을 지원하면서 추가적으로 사용되지 않는 메모리 셀들을 활용하는 방법을 제안한다.

1. 기본 함수들

동일한 메모리 주소 상에서 사용되지 않는 메모리 모듈들을 메모리 주소가 증가함에 따라 연속적으로 인접하도록 배치하기 위해서 메모리 모듈을 할당하는 수식(1)을 수식(3)로, 그에 대응하는 주소계산 및 할당하는 수식(2)를 수식(4)로 수정하였다.

$$\grave{\mu}(x,y)=(px+y)\%m,\ 0\le x<M,\ 0\le y<N. \quad (3)$$

$$\grave{\alpha}(x,y)=x/q+(y/p)\cdot s1,\ 0\le x<M,\ 0\le y<N. \quad (4)$$

여기서 $s1$는 $\lceil M/q \rceil$인 음이 아닌 정수이다.

간략한 예로써 $M\times N=20\times 20$, $p=q=2$, 그리고 $m=5$일 경우를 고려해 볼 때, 수식(1)에 의해 계산된 메모리 모듈들이 할당된 결과는 [그림 1]-(c)이며 수식(3)의 결과는 [그림 1]-(d)와 같다. $p=q$인 경우에는 주소계산 및 할당 함수들인 수식(2)와 수식(4)가 동일한 결과를 갖으며 [그림 1]-(b)와 같다.

계산된 주소값에 의해 각각의 메모리모듈할당함수들에 의해 데이터들이 할당된 결과는 [그림 1]-(e)과 같다. 여기서 동일한 메모리 주소 상에서 사용되지 않는 메모리 모듈들의 셀은 'x'로 표시하였다. [그림 1]-(e)의 오른쪽 메모리 모듈들의 내용을 살펴보면 메모리 주소가 증가함에 따라 사용되지 않는 메모리 모듈들이 연속해서 인접해 있다.

2. 메모리 모듈 선택

수정된 메모리모듈할당함수($\grave{\mu}(x,y)$)를 토대로 접근형태별로 m개의 메모리 모듈들 중에서 n개를 선택하는 함수들은 다음과 같다.

BLK: $\mu 1(k)=(\mu 1(0)+rk)\%m$,
ROW: $\mu 1(k)=(\mu 1(0)+rpk)\%m$,
COL : $\mu 1(k)=(\mu 1(0)+rk)\%m$,
FRD: $\mu 1(k)=(\mu 1(0)+rqk)\%m$,
BKD: $\mu 1(k)=(\mu 1(0)+rk)\%m, 0\le k<pq.$

여기서 $\mu 1(0)=\grave{\mu}(x,y)$.

3. 주소 계산

수정된 주소계산 및 할당함수($\grave{\alpha}(x,y)$)를 토대로 m개의 메모리 모듈들 중에서 선택된 n개에 대한 주소들을 계산하는 접근형태별 함수들은 다음과 같다.

BLK: $\alpha 1(k)=\grave{\alpha}(x+(rk)/p, y+(rk)\%p)$,
ROW: $\alpha 1(k)=\grave{\alpha}(x+rk, y)$,
COL: $\alpha 1(k)=\grave{\alpha}(x, y+rk)$,
FRD: $\alpha 1(k)=\grave{\alpha}(x+rk, y+rk)$,
BKD: $\alpha 1(k)=\grave{\alpha}(x+rk, y-rk),\ 0\le k<n.$

4. 라우팅

II-3절의 내용과 동일한 기능을 수행하기 위해서 입력된 데이터와 계산된 주소들은 다음과 같이 접근형태별로 라우팅을 수행한다.

BLK, COL, BKD: $RO(rk)=RI(k)$,
ROW: $RO((prk)\%m)=DI(k)$,
FRD: $RO((qrk)\%m)=RI(k),\ 0\le k<pq.$

데이터와 계산된 주소들을 접근형태별로 상기 함수들에 의해 각각 재정렬시킨 후 $\grave{\mu}(x,y)$만큼 right-shift를 수행한다. 상기 과정은 쓰기 연산인 경우이며, 읽기 연산의 경우는 상기 기술된 과정들을 역순으로 수행한다.

5. 지역 버퍼 메모리 시스템

병렬 프로그램의 관점에서 볼 때, 사용 가능한 메모리의 크기인 $M\times N$ 배열 이외의 사용되지 않는 메모리 셀들에 대한 직접적인 접근을 고려하기는 어렵다. 그러나 시스템을 설계하면서 다중접근기억장치 내의 사용

되지 않는 메모리 셀들을 처리기들이 직접적으로 접근할 수 있는 지역 버퍼 메모리 시스템으로 재구성하여 활용할 수 있다. 이 경우는 Ⅱ-2 절과 Ⅱ-3절에서 언급된 기능들은 고려의 대상에서 배제된다.

사용하지 않는 메모리 셀들을 활용하기 위한 추가적인 접근은 다중접근기억장치에서 제공하는 n개를 기본 단위로 n개의 $(N/p)\times t$배수만큼 지원할 수 있다. 이를 위한 추가된 메모리모듈할당함수 $M(i,k)$는 다음과 같다.

$$M(i,k) = (M(i) + k\%pq)\%m.$$
$$M(i) = \grave{\mu}(s2 \cdot q,\ pi) = (s2 \cdot pq + pi)\%m.$$

여기서 변수 i와 k의 범위는 각각 $0 \leq i < N/p$, $0 \leq k < n \times t$이고, $s2$는 M/q이다. 그리고 t는 사용되지 않는 메모리 셀들이 메모리 모듈들 간에 연속적으로 나열되는 개수를 n으로 나눈 결과로써 다음과 같이 정의된다.

$$t = \begin{cases} (c-n)/m+1 & c \geq n \\ 0 & c < n \end{cases}$$

여기서 c는 $s2\times(m-n)$이다.

$s2$가 n보다 충분히 커야 사용되지 않는 메모리 셀들 중에서 n개의 데이터를 동시에 접근할 수 있다. 반면에 $t=0$일 경우는 사용되지 않는 메모리 셀들이 n개 이상 존재하지만 메모리 모듈들에 접근할 때 충돌이 생겨 동시에 n개의 데이터에 접근할 수 없음을 의미한다.

그리고 추가된 메모리모듈할당함수에 해당하는 추가된 주소계산 및 할당을 위한 함수 $A(i,k)$는 다음과 같다.

$$A(i,k) = A(i) - (k/(m-n)) - (k/n).$$
$$A(i) = \grave{\alpha}(M, pi) - 1 = s2 + s1 \cdot i - 1.$$

여기서 변수 i와 k의 범위는 각각 $0 \leq i < N/p$, $0 \leq k < n \times t$이다.

상기 추가된 메모리모듈할당함수와 주소계산 및 할당함수로 병렬프로그램이 기존의 다중접근기억장치에서 $M \times N$만큼의 메모리 셀들을 활용하면서 추가적으로 처리기들이 $(n \times t) \times N/p$만큼의 메모리 셀들을 지역 버퍼 메모리로 활용할 수 있다. [표 1]은 기존의 다중접근기억장치의 메모리 활용도와 추가적인 메모리 활용이 포함된 전체 메모리의 활용도를 비교한 것이다.

[표 1]에서 *Type A*는 $M \times N = 1920 \times 1280$이고, *Type B*

표 1. 설계 파라메터에 따른 메모리 용량 및 활용도

p=q	m	m-pq	M x N	MemCap	RU(%)	t	ExtUse	TotUse	RTU(%)
2	5	1	type A	3072000	80.00	192	491520	2949120	96.00
2	5	1	type B	3080005	79.90	192	491520	2952321	95.85
3	11	2	type A	3006080	81.75	116	444744	2902344	96.55
3	11	2	type B	3010777	81.73	116	445788	2906589	96.54
5	29	4	type A	2850816	86.21	53	339200	2796800	98.11
5	29	4	type B	2869405	85.76	53	339200	2800001	97.58
6	37	1	type A	2533760	96.99	8	61344	2518944	99.42
6	37	1	type B	2541678	96.82	8	61344	2522145	99.23
11	127	6	type A	2600325	94.51	8	112288	2569888	98.83
11	127	6	type B	2600325	94.63	8	112288	2573089	98.95
12	149	5	type A	2550880	96.34	5	76320	2533920	99.34
12	149	5	type B	2566823	95.87	5	76320	2537121	98.84
15	227	2	type A	2498816	98.35	1	19125	2476725	99.12
15	227	2	type B	2518338	97.72	1	19125	2479926	98.47
16	257	1	type A	2467200	99.61	0	0	2457600	99.61
16	257	1	type B	2518857	97.70	0	0	2460801	97.70
18	331	7	type A	2550024	96.38	2	46008	2503608	98.18
18	331	7	type B	2550024	96.50	2	46008	2506809	98.31
19	367	6	type A	2545512	96.55	1	24187	2481787	97.50
19	367	6	type B	2545512	96.67	1	24187	2484988	97.62
20	401	1	type A	2463744	99.75	0	0	2457600	99.75
20	401	1	type B	2528305	97.33	0	0	2460801	97.33

는 $M \times N = 1921 \times 1281$이다. 그리고 $MemCap$은 $M \times N$을 저장하기 위해 기존의 다중접근기억장치가 필요한 실제 메모리 용량이며, $RU(=(M \times N)/MemCap \times 100\%)$의 메모리 활용도를 나타낸다. 그리고 $ExtUse$는 본 논문에서 제안한 방법에 의해 추가되는 메모리 용량이며, $TotUse$는 $(M \times N) + ExtUse$이다. 본 논문에서 제안하는 추가 활용방법을 적용하였을 경우 실제 메모리의 총 활용도는 $RTU(= TotUse/MemCap \times 100\%)$가 된다.

실험을 통해서 $p = q$가 2와 3, 5와 6, 그리고 16, 18, 19, 20인 경우를 각각 비교했을 때 $m-n$가 작을수록 메모리의 활용도가 높고, $p = q$ 열과 $M \times N$ 열을 관련지어 볼 때 $M\%p$와 $N\%q$가 작을수록 메모리 활용도가 높다. 또한 $m-n=1$인 $p=q$가 2, 6, 16, 20인 경우와 $m-n=2$인 $p=q$가 3과 15인 경우를 각각 서로 비교해 보면 n값이 클수록 메모리의 활용도가 높다.

$t > 0$인 경우에 본 논문에서 제안한 방법이 기존의 방법보다 추가적으로 $(n \times t) \times N/p$ 만큼의 메모리 셀들을 더 사용함으로써 메모리 활용도를 향상시켰고 각각의 처리기들이 $t \times N/p$ 만큼의 메모리를 지역 버퍼 메모리로써 활용할 수 있다.

IV. 결론

고성능의 컴퓨팅을 위한 병렬처리시스템에서는 프로세서와 메모리시스템의 구조 및 이들을 활용할 프로그램의 병렬성이 고려되어야 한다. 이 중 방대한 양의 데이터에 대한 전송률을 확보하기 위해서는 메모리시스템에 대한 연구가 필수적이다.

선형 비틀림 구조를 갖는 다중접근기억장치는 m개의 메모리 모듈들에 다양한 접근형태들과 임의의 간격을 허용하면서 임의의 좌표를 기준으로 동시에 n개의 데이터에 접근할 수 있으며 SIMD 컴퓨터에 적합한 메모리 시스템으로 간주된다. 하지만 다중접근기억장치는 일반적으로 $(m-n) \times (\dot{\alpha}(M-1, N-1)+1)$만큼의 메모리 셀들을 활용하지 않는다.

본 논문에서는 $t > 0$인 조건에서 사용되지 않는 메모리 셀들 중 $(n \times t) \times N/p$ 만큼을 처리기들이 n개 단위로 동시에 접근함으로써 처리기들의 지역 버퍼 메모리 시스템으로써의 활용을 제안하였다.

앞에서 언급했던 바와 같이 다중접근기억장치는 메모리 모듈들을 포함하여 하나의 칩으로 직접되는 것이 효율적이다. 메모리모듈들의 수가 클수록 이에 따른 데이터 및 주소 버스들과 제어신호들이 기하급수적으로 증가하기 때문이며 집적되는 회로의 공간 복잡도 보다 핀 아웃의 개수에 따른 제한사항이 더 크기 때문이다. 그래서 메모리 모듈들을 포함한 다중접근기억장치가 하나의 칩에 집적된다면 CUDA 구조[1]에서처럼 다중접근기억장치는 공유 메모리로써, 추가된 지역 버퍼 메모리는 지역 메모리로써 활용될 수 있을 것이다. 이는 다중접근기억장치와 연계된 처리기들이 내부적으로 다수의 레지스터들을 갖겠지만 동일한 환경 하에서 추가적으로 좀 더 많은 메모리를 확보할 수 있다면 프로그램 측면에서는 좀 더 유연하게 수행될 수 있으며 처리속도 또한 높일 수 있다.

참 고 문 헌

[1] D. B. Kirk and Wen-mei W. Hwu, *Programming Massively Parallel Processors: A Hands-on Approach*, Morgan Kaufmann Pub., 2010.

[2] T. G. Mattson, B. A. Sanders, and B. L. Massingill, *Patterns for Parallel Programming,* Addison-wesley Pub., 2005.

[3] D. T. Harper III. "Block, Multistride Vector, and FFT Accesses in Parallel Memory System," IEEE Trans. Parallel and Distributed Systems, Vol.2, No.1, pp.43-51, 1991(1).

[4] R. Raghavan and J. P. Hayes, "On Randomlu Interleaved Memories," Proc. Supercomputing '90, pp.49-58, 1990.

[5] N. B. MacDonald, "An Overview of SIMD Parallel System: AMT DAP, Thinking Machines CM-200, and MasPar MP-1," Proc. Workshop Parallel Computing, 1992(4).

[6] K. Kim and V. K. P Kumar, "Perfect Latin Squares and Parallel Array Access," Pro. Int'l Symp. Computer Architecture, pp.372-379, 1989.

[7] D. T. Harper III. "A Multiacess Frame Buffer Architecture," IEEE Trans. Computers, Vol.43, pp.618-622, 1994(5).

[8] P. Budnik and D. J. Kuck, "The Organization and USe of Parallel Memories," IEEE Trans. Computers, Vol.20, No.12, pp.1566-1569, 1971(12).

[9] D. H. Lawire, "Access and Alignment of Data in an Array Processor," IEEE Trans. Computers, Vol.24, No.12, pp.1145-1155, 1975(12).

[10] D. C. Van Voorhis and T.H.Morrin, "Memory System for Image Processing," IEEE Trans. Computers, Vol.27, No.2, pp.113-125, 1978(2).

[11] D. H. Lawrie and C. R. Vora, "The Prime Memory System for Array Access," IEEE Trans. Computers, Vol.31, No.5, pp.435-442, 1982(5).

[12] J. W. Park, "An Efficient Buffer Memory System for Subarray Access," IEEE Trans. Parallel and Distributed Systems, Vol.12, No.3, pp.316-335, 2001(3).

[13] Eero Aho, Jarno Vanne, and T. D. Hamalainen, "Parallel Memory Architecture for Arbitrary Stride Access," In Proc. of the IEEE Workshop on Design and Diagnostics of Electronic Circuits and System, pp.65-70, 2006(4).

[14] J. K. Tanskanen and T. Pitkanen, "Parallel Memory Architecture for TTA Processor," Proc. of the 7th Int. Conf. on Embedded Computer Sys.: Architecture Modelling, and Simulation," pp.273-283, 2007.

[15] Dionysios Reisis and Nikolaos Vlassopoulos, "Conflict-free Parallel Memory Accessing Techniques for FFT Architecture," IEEE Trans. on Circuits and Systems, Vol.55, No.11, pp.3438-3447, 2008(12).

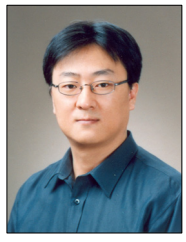
[16] J. W. Park, "Multiaccess Memory System for Attached SIMD Computer," IEEE Trans. Computers, Vol.53, No.3, pp.1-14, 2004(3).

[17] H. Lee and J. W. Park, "Parallel Processing System for Multi-Access Memory System," Proc. World Multi-conference of Systematics, Cybernetics, and Informatics, pp.561-565, 2000.

[18] H. Lee, H. K. Cho, D. S. You, and J. W. Park, "MAMS-PP4: Multi-Access Memory System used to improve the processing speed of Visual Media Applications in a Parallel Processing System," IEICE Trans. Fundamentals of Electronics Communications and Computer Sciences, Vol.E87-A, No.11, pp.2852-2858, 2004(11).

저 자 소 개

이 형(Hyung Lee) 정회원



- 1997년 2월 : 충남대학교 컴퓨터공학과(공학석사)
- 2005년 2월 : 충남대학교 컴퓨터공학과(공학박사)
- 2001년 3월 ~ 현재 : 대전보건대학 방송제작과 조교수

<관심분야> : 영상/비디오처리, 디지털방송