# 분산 공유 메모리 시스템에서 메모리 접근지연을 줄이기 위한 이중 슬롯링 구조

민 준 식† · 장 태 무††


## 요 약

집적회로 기술의 발달은 처리기의 속도를 계속적으로 증가시켜 왔다. 처리기 응용분야의 주요한 도전은 공유 메모리 다중 처리기 시스템에서 고성능 처리기들을 효과적으로 사용하고자 하는 것이다. 우리는 상호 연결망 문제가 소규모의 공유 메모리 다중처리기 시스템에서 조차 완전히 해결되었다고 생각하지 않는다. 그 이유는 공유버스의 속도는 새로운 강력한 처리기들의 대역폭 요구를 수용할 수 없기 때문이다. 지난 수년간 점대점 단방향 연결은 매우 가능성 있는 상호 연결망 기술로서 대두되었다. 단일 슬롯링은 점대점 상호 연결망의 가장 간단한 형태이다. 단일 슬롯링 구조의 단점은 링에서 처리기의 수가 증가함에 따라 메모리 접근지연 시간이 선형적으로 증가한다는 것이다. 이런 이유로 우리는 캐쉬 기반의 다중처리기 시스템에서 단일 슬롯링을 대체할 수 있는 이중 슬롯링 구조를 제안한다. 또한 본 논문에서 새로운 스누핑 프로토콜을 사용하는 이중 슬롯링 구조를 분석하고 분석적모델과 모의 실험을 통하여 기존의 단일 슬롯링과 성능을 비교한다.


# A Dual Slotted Ring Organization for Reducing Memory Access Latency in Distributed Shared Memory System

JunSik Min† · TaeMu Chang††


## ABSTRACT

Advances in circuit and integration technology are continuously boosting the speed of processors. One of the main challenges presented by such developments is the effective use of powerful processors in shared memory multiprocessor system. We believe that the interconnection problem is not solved even for small scale shared memory multiprocessor, since the speed of shared buses is unlikely to keep up with the bandwidth requirements of new powerful processors. In the past few years, point-to-point unidirectional connection have emerged as a very promising interconnection technology. The single slotted ring is the simplest form point-to-point interconnection. The main limitation of the single slotted ring architecture is that latency of access increase linearly with the number of the processors in the ring. Because of this, we proposed the dual slotted ring as an alternative to single slotted ring for cache-based multiprocessor system. In this paper, we analyze the proposed dual slotted ring architecture using new snooping protocol and enforce simulation to compare it with single slotted ring.


**키워드 : 캐쉬일관성(Cache Coherence), 상호연결망(Interconnection Network), 병렬처리(Parallel Processing)**

## 1. 서 론

오늘날 병렬처리 시스템은 고성능 계산을 위한 가능성 있는 대안으로 대두되었다[4]. 현재의 대부분의 고성능 컴퓨터는 공유 또는 분산 메모리 다중 컴퓨터로서 수십에서 수백개의 처리기들을 탑재할 수 있다. 비록 수백개의 처리기들은 연결할 수 있는 상호 연결망(Interconnection Network)에 대한 많은 연구들이 진행되고 있지만 소규모의 공유 메모리 처리기들을 연결하기 위한 상호 연결망은 여전히 숙제로 남아 있다. 새롭고 점점 빠른 처리기들이 매년 등장함에 따라 현재 상용 시스템에서 널리 사용되는 공유버스는 처리기들을 연결하는 상호 연결망으로서 부적합하다. 버스가 상호 연결망으로서 확장성에 제약을 받는 주요 이유는 다음과 같은 버스 자체가 가지고 있는 특성 때문이다[3]. 첫째, 버스는 특정한 시간에 단지 하나의 처리기만 전송 권한을 가지는 상호 배타적인 자원이다. 둘째, 모든 처리기는 버스에 접근하기 전에 중재가 이루어져야 한다. 셋째, 버스 클럭 사이클(Clock Cycle)

† 준 회 원 : 한국전산원 국가망 이용관리 팀장
†† 정 회 원 : 동국대학교 컴퓨터 · 멀티미디어공학과 교수
논문접수 : 2001년 4월 16일, 심사완료 : 2001년 10월 19일

은 신호가 버스 전체에 전달될 수 있도록 길어야 한다. 한 버스 클럭 사이클 동안에 보다 많은 자료를 전송하기 위하여 버스의 폭을 넓히는 것이 하나의 방법이지만 버스의 핀 수나 상호 간섭 등으로 버스의 폭을 넓히는데 한계가 있다.

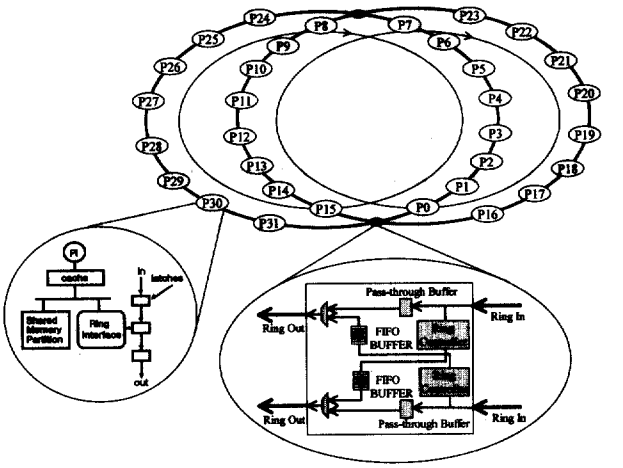
지난 수년간 점대점(Point-to-Point) 단방향(Unidirectional) 연결은 버스를 대체할 수 있는 가능성 있는 상호 연결망 기술로서 대두되었다. 점대점 연결은 묶음(Packaging) 기능을 가진 전송기(Transmitter)와 수신기(Receiver)가 있으며 신호는 중첩 되어진다. 또한 새로운 전송은 이전 전송이 수신기에 도착하기 전에 점대점 연결에서 시작될 수 있기 때문에 클럭 스피드와 처리량(Throughput)은 연결선(Wire) 길이에 제약을 받지 않는다. 전체적으로 상호 연결망은 버스보다 기술적으로 좋은 확장성(Scalability)을 가진다. 점대점 통신의 잠재성은 IEEE의 SCI(Scalable Coherent Interface)에 의해서 증명 되었다[11]. 슬롯링은 점대점 단방향 상호 연결망의 가장 간단한 형태중의 하나로서 제안되었다[17, 18]. 이런 단방향 슬롯링은 버스에 비하여 상대적으로 큰 대역폭을 지니기 때문에 많은 처리기들을 연결할 수 있으나 처리기수가 증가함에 따라 링의 길이가 증가하게 된다. 따라서 링 길이의 증가는 시스템 내의 구성 요소들간에 통신하기 위한 시간이 점점 길어져 성능의 감소로 이어진다[19].

본 논문에서는 슬롯링[2]을 상호 연결망으로 하는 시스템에서 처리기수가 증가할 경우 링 길이의 증가로 인한 메모리 접근지연(Access Latency) 시간의 증가로 시스템의 성능이 감소하는 단점을 개선하기 위한 이중 슬롯링(Dual Slotted Ring) 구조 및 캐쉬 일관성 프로토콜을 제안한다. 또한 기존의 단일 슬롯링과 새로 제안된 이중 슬롯링 구조의 시스템을 분석적 모델과 실행기반의 모의실험을 통하여 성능을 비교함으로서 이중 슬롯링 구조를 채택한 시스템이 보다 성능이 우수함을 확인한다

## 2. 이중 슬롯링 구조

슬롯링 구조는 공유버스를 대체할 수 있는 점대점 상호 연결망의 한 종류로서 제안되었다. 기존에 제안된 단일 슬롯링 구조는 상호 연결망으로서 공유 버스의 대안으로 가능성을 보여 주었으나 시스템의 성능을 높이기 위하여 많은 처리기들을 탑재할 경우 메모리 접근지연 시간이 증가하는 단점을 지닌다. 본 논문에서는 이러한 단점을 개선하기 위하여 각 처리기들을 연결하는 링을 중첩한 형태의 이중 슬롯링 구조를 제안한다.

이중 슬롯링은 점대점 상호 연결망인 단방향 링의 한 종류로서 (그림 1)과 같이 노드들로 구성되며 한 방향으로만 자료를 전송한다. 이중 슬롯링 구조는 단일 링과 비교하여 처리기 수는 동일하지만 두개의 링에 배치하여 링의 길이를 짧게 함으로서 메모리 접근지연 시간을 줄인다. 또한 링에는 통신을 하기 위한 두개의 연결노드(Interconnection Node)를 가지며 이러한 연결노드를 통하여 링간에 통신을 하게 된다. 연결 노드를 여러 개로 할 경우 링간 통신전달 지연시간을 줄일 수 있으나 하드웨어의 복잡성 및 비용을 고려하여 두개로 설계하였다. 단방향 슬롯링에서는 가능한 간단한 경로 결정(Routing) 매커니즘(Mechanism)이 요구된다. 슬롯링에서의 경로 결정은 단지 현재 노드에서 다음 노드로의 전송 여부만 결정하면 된다. 이것은 통신 지연을 최대한 줄이고 점대점 연결에서 제공되는 원천적인 대역폭을 최대한 이용하기 위함이다.




(그림 1) 이중 슬롯링의 구성

각각의 노드는 지역캐쉬(Local Cache)에 연결된 처리기, 메모리 모듈과 링 접속기(Ring Interface) 등으로 구성되어 있다. 또한 링에서의 전송은 동일한 클럭에 의해서 동기화되며, 각각의 처리기 연산에 대해서는 비동기적이다. 그림에서 볼 수 있듯이 공유 메모리는 링에 있는 각각의 노드에 분산되어 있으며, 시스템에 있는 모든 처리기들은 하나의 동일한 주소영역을 사용하고, 블록이 사상(Mapping)되는 노드를 홈 노드(Home Node)라 한다. 하나의 더티 비트(Dirty Bit)가 현재 메모리 블록이 가장 최근 것인지의 여부를 나타내기 위하여 사용된다

## 3. 캐쉬 일관성 프로토콜

### 3.1 개요

공유 메모리 다중 처리기 시스템에서 여러 캐쉬들에 의해서 공유되는 블록들간에 일관성을 유지하기 위해서 하드웨어로 구현되는 프로토콜은 크게 두 종류로 나누어진다. 하나는 공유 버스를 기반으로 하는 시스템과 같이 메모리 참조 요구

를 공유 버스를 통하여 방송(Broadcasting)하며, 일관성은 각각의 캐쉬가 버스를 스누핑함에 의해서 유지되는 스누피 프로토콜(Snoopy Protocol)[6]이다. 다른 하나는 상대적으로 방송이 불리한 다단계 상호 연결망(Multistage Interconnection Network)에서와 같이 일관성을 유지하기 위하여 공유되는 캐쉬 블록에 대한 정보를 메모리에 별도로 유지하는 디렉토리를 기반으로 하는 방법(Directory-based Scheme)[7]이다.

다른 점대점 상호 연결망과[12, 13]는 대조적으로 슬롯링 구조에서는 방송이 가능하고 메모리 접근 지연시간 측면에서 유리하며, 효율성이 좋고 간단한 디렉토리 유지로 인해서 메모리 낭비가 없는 스누핑 프로토콜이 적합하다[3]. 슬롯링을 상호 연결망으로 하는 시스템에서 채택된 기존의 스누핑 프로토콜[2]은 지연쓰기[Write Back]와 기록 무효화(Write Invalidation)[1]를 바탕으로 한다[5]. 본 논문에서 제안하는 이중 슬롯링 구조에서는 캐쉬에 있는 수정된 블록(Dirty)이 다른 캐쉬에 의해서 공유될 때 메모리를 갱신하는 지연쓰기 정책을 사용하는 기존의 프로토콜과는 달리 새로운 스누핑 프로토콜을 제안한다. 새로운 프로토콜의 경우에 수정된 블록이 다른 캐쉬에 의해서 공유되면 메모리를 갱신하지 않고 블록의 소유권을 캐쉬가 유지하며 이 블록에 대한 참조요구가 있을시 블록을 제공하는 역할을 한다. 이 경우에 메모리 블록이 갱신되는 경우는 해당 블록이 캐쉬에서 대치가 될 때 일어나게 된다. 새로운 프로토콜에서는 동일한 메모리 블록의 사본들간에 일관성을 유지하기 위하여, 캐쉬 블록은 Invalid(Inv), Read Shared(RS), Write Exclusive(WE), Modified Shared(MS) 상태 중의 하나를 유지한다. 블록은 실제 응용프로그램이 실행되는 동안에 각 처리기의 메모리 참조 요구에 따라 일관성을 유지하며 상태를 변환한다

### 3.2 일관성 유지를 위한 상태 분석

응용프로그램이 실행되는 동안에 일관성 동작은 읽기 실패(Read Miss), 쓰기 실패(Write Miss), 쓰기 적중(Write Hit), 블록 대치(Replacement)의 경우에 대해서 필요하다. 메모리 블록의 복사본은 다음의 상태중 하나를 유지하며 처리기의 메모리 참조 요구에 따라서 적절하게 상태를 변환함으로써 일관성을 유지한다.

① Invalid(INV) : 일관성이 없거나 캐쉬에 존재하지 않는 상태

② Read Shared(RS) : 일관성이 있으며 동일한 사본이 다수 존재할 수 있는 상태

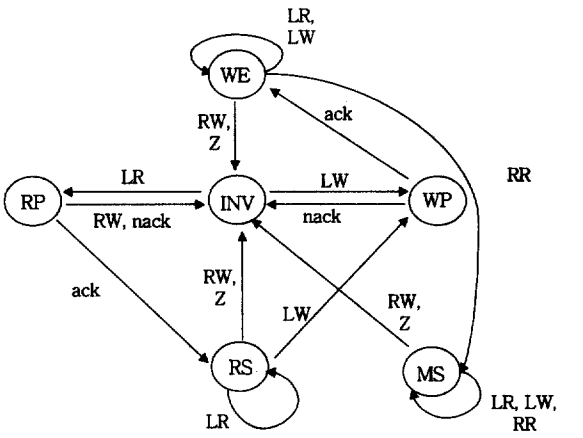
③ Write Exclusive(WE) : 해당 블록은 수정되어 메모리 블록과 일관성이 없으며, 시스템 내에서 유일하게 정확성을 유지하는 상태

④ Modified Shared(MS) : 해당 블록은 수정되어 메모리 블록과 일관성이 없으며, 캐쉬들간에 공유되어 있는 상태

공유 버스를 상호 연결망으로 하는 시스템에서는 블록들간에 일관성을 유지하기 위한 상태 변환시에 버스에 중재기(Arbitrator)가 있어서 일관성 유지를 필요로 하는 사건(Event)들 사이에 중재를 담당하므로 충돌이 일어나지 않는다. 그러나 슬롯링을 상호 연결망으로 하는 다중 처리기 시스템에서는 공유 버스를 기반으로 하는 시스템과는 달리 메모리 참조 요구를 가진 여러 개의 메세지가 특정한 시간내에 동시에 링을 순환할 수 있다. 이 때에 하나 이상의 노드가 동시에 같은 블록에 대해서 메모리 참조 요구를 할 경우 충돌이 일어나게 되며, 이를 해결할 수 있는 적절한 방법이 요구된다.

링 프로토콜에서는 Read Pending(RP)과 Write Pending(WP)의 두 가지 부가적인 상태로서 이러한 충돌을 해결한다. 노드간에 메모리 참조 요구가 충돌이 일어날 경우는 어느 한 노드만 상태 변환, 즉 메모리 참조 요구가 허락되며 나머지 참조 요구들은 상태 변환이 허락되지 않으며 참조 요구를 재시도 해야만 한다. 이러한 Pending 상태는 실제 캐쉬 디렉토리에는 존재하지 않는 상태이며, 스누퍼(Snooper)에 있는 레지스터에 블록의 참조 요구가 Pending 상태에 있다는 정보만 유지하게 된다.

Pending상태에 있는 캐쉬 블록은 그 블록의 참조 요구에 대한 응답을 아직 받지 않은 상태이며 블록 미스에 대한 응답은 블록을 포함하고 있는 메세지이다. 만약 Pending 상태에 있는 블록들이 요구에 대한 긍정응답(Acknowledgment)을 받지 못하면 상태 변환을 포기하고 그 블록에 대한 참조 요구를 재시도해야만 한다. (그림 2)는 새로운 슬롯링 프로토콜에서 캐쉬에 있는 블록들의 상태 변환을 나타내고 있다.




(그림 2) 새로운 프로토콜의 캐쉬블록 상태 전이도

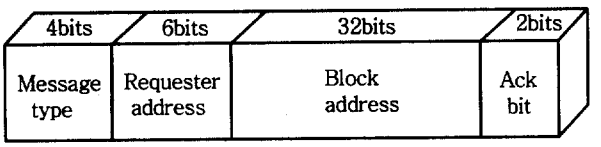
### 3.3 일관성 유지를 위한 메세지

슬롯링에서의 프로토콜은 일관성을 유지하기 위해서 슬롯링을 통하여 전달되는 메세지의 집합으로 구현될 수 있다. 각 노드의 물리적 주소공간에 사상되는 캐쉬 블록 미스는 지역 미스(Local Miss)라고 한다. 이 경우에 읽기 실패에 대해서는 해당 블록의 메모리가 변경되지 않았으면 일관성을 유지하기 위한 별도의 메세지를 필요로 하지 않으며, 그 밖의 실패나 쓰기 적중, 블록 대치는 일관성을 유지하기 위하여 링을 통하여 메세지를 전달하게 된다.

#### 3.3.1 메세지 형태

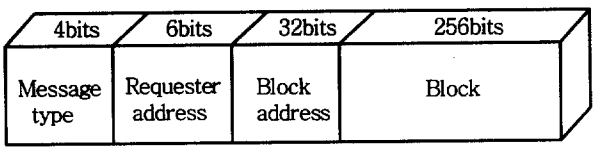
메세지 형태는 다음과 같은 2가지 종류로 나누어진다.

① 조회(Probe) 메세지 : 메모리 참조 요구나 일관성 유지를 위한 짧은 형태의 메세지로서, Read-block, Write-block, Write-hit 등이 있다. 다음은 내부 구성 형태이다.




(그림 3) 조회 메세지 구성도

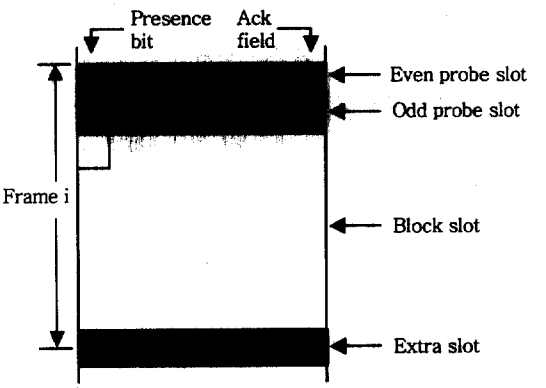
② 블록(Block) 메세지 : 조회 메세지에 대한 응답을 하기 위해서 블록을 포함하고 있는 긴 형태의 메세지이다. Send-block 메세지가 여기에 해당된다.




(그림 4) 블록 메세지 구성도

조회 메세지는 슬롯링을 통해 전송하기 위해서 단지 하나의 64bits 패킷 슬롯(Packet Slot)과 하나의 파이프라인 단계를 필요로 한다. 반면에 블록 메세지는 5개의 연속적인 패킷 슬롯과 파이프라인 단계를 필요로 한다. 이러한 두 종류의 메세지는 링에서 조회 슬롯(Probe Slot)과 블록 슬롯(Block Slot)에 실리게 되며, 슬롯들이 모여서 하나의 프레임(Frame)을 구성하게 된다.

각각의 프레임은 짝수 주소 블록에 대한 짝수 조회 슬롯(Even Probe Slot)과 홀수 주소 블록을 위한 홀수 조회 슬롯(Odd Probe Slot), 블록 슬롯, 그리고 노드간에 인터럽트(Interrupt)신호를 위한 하나의 부가적인 슬롯(Extra Slot)으로 구성된다. (그림 5)은 프레임의 구성을 나타낸 것이다.




(그림 5) 프레임의 구성도

## 4. 프로토콜의 정당성

일반적인 다중 처리기 시스템에서는 다수의 프로세스(Process)가 동일한 기억장치의 영역에 접근하므로 프로그램의 정확한 수행을 위하여 사건 순서 모형(Event Ordering Model)이 필요하다. 그 중 가장 엄밀한 모형이 Lamport의 순차 일치성(Sequential Consistency)[14]이며, Scheurich와 Dubois는 캐쉬가 기반이 된 시스템에서 순차 일치성의 충분조건을 다음과 같이 제시하였다[15].

① 모든 처리기는 프로그램에서 명시된 순서대로 기억장치에 접근한다.

② 처리기는 이전의 접근들이 전역적으로 수행되지(Globally Performed) 않으면 새로운 접근을 시작하지 않는다.

여기서 쓰기가 전역적으로 수행되었다 함은 수정된 상황이 모든 처리기에게 알려짐을 의미하며, 읽기의 경우는 돌려 받는 값이 정해져 있고 그 값을 수정한 쓰기가 전역적으로 수행된 것을 의미한다.

본 논문에 제시된 프로토콜의 경우,

① 모든 처리기는 프로그램에서 명시된 순서대로 기억장치에 접근한다.

② 쓰기의 경우에 블록의 상태가 RS일 때 동작 이전에 무효화 신호를 방송하여 모든 캐쉬에게 알리며, 캐쉬에 존재하는 동일한 블록들을 무효화한다.

③ 읽기의 경우에도 각각의 캐쉬는 링을 관찰하고 있고, 블록의 상태가 일관성이 있으므로 이전의 쓰기가 완료된 MS나 WE로 된 상태의 블록을 읽어가면 만족한다.

이와 같이 순차 일치성을 만족하므로 본 논문의 프로토콜로서 병렬 프로그램을 안전하게 실행함을 확인할 수 있다.

## 5. 성능평가

본 논문에서는 성능평가를 위하여 두 가지 방법을 사용했

다. 첫 번째 방법은 분석적 모델을 통하여 성능평가를 시행하고 두 번째 방법은 실제 상황을 반영하기 위하여 실행구동(Exectuion-driven)방식의 모의실험을 시행하였다.

### 5.1 분석모델

메모리 접근지연 시간은 시스템의 성능을 결정하는 매우 중요한 요소중의 하나이다. 슬롯링을 상호연결망으로 하는 공유 메모리 다중처리기 시스템에서 스누핑 프로토콜을 사용하는 경우에 메모리 접근지연 시간은 블록을 요구하는 조회 메세지 전달시간과 요구한 블록을 전달받기 위한 블록 메세지 전달시간으로 나뉘어진다. 노드수가 동일할 경우에 본 논문에서 제안한 이중 슬롯링 구조는 링을 중첩하여 설계함으로써 기존의 단일 슬롯링 구조보다 메세지 전달 경로를 줄일 수 있다. 메세지 전달 경로가 줄어듦은 메세지 전달 지연시간이 짧아짐을 의미한다.

<표 1>은 단일링과 본 논문에서 제시된 이중링의 메세지 전달지연 시간을 비교 분석한 것으로서 모든 경우에 있어서 새로 제안된 이중 슬롯링 구조가 기존의 단일링 구조보다 메세지 전달지연 시간이 동일하거나 짧음을 알 수 있다.

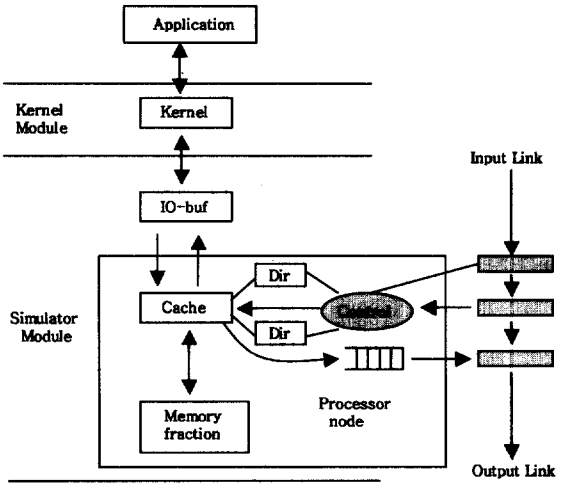
〈표 1〉 메세지 전달지연 시간 비교

| 구 분 | | | 메세지 전달 지연 시간 |
|---|---|---|---|
| 조회 메세지 | 단일링 | 최 소 | $T_R \times H_N + T_{req}$ |
| | | 최 대 | $T_R \times H_N + T_{req}$ |
| | 이중링 | 최 소 | $T_R \times H_{(N/2)} + T_{req}$ |
| | | 최 대 | $T_R \times H_{(3N/4-1)} + T_{req}$ |
| 블록 메세지 | 단일링 | 최 소 | $T_D \times H_1 + T_{lookup} + T_{data}$ |
| | | 최 대 | $T_D \times H_{N-1} + T_{lookup} + T_{data}$ |
| | 이중링 | 최 소 | $T_D \times H_1 + T_{lookup} + T_{data}$ |
| | | 최 대 | $T_D \times H_{(3N/4-1)} + T_{lookup} + T_{data}$ |

- $T_R$ : 조회 메세지가 다음 노드로 이동하기 위한 시간
- $T_D$ : 블록 메세지가 다음 노드로 이동하기 위한 시간
- $T_{req}$ : 조회 메세지를 링에 삽입하기 위하여 대기하는 시간
- $T_{data}$ : 블록 메세지를 링에 삽입하기 위하여 대기하는 시간
- $T_{lookup}$ : 메모리 상태를 조회하기 위한 시간
- $H_N$ : 메세지가 순환하는 경로(hops)의 수(N은 시스템내 노드의 수)

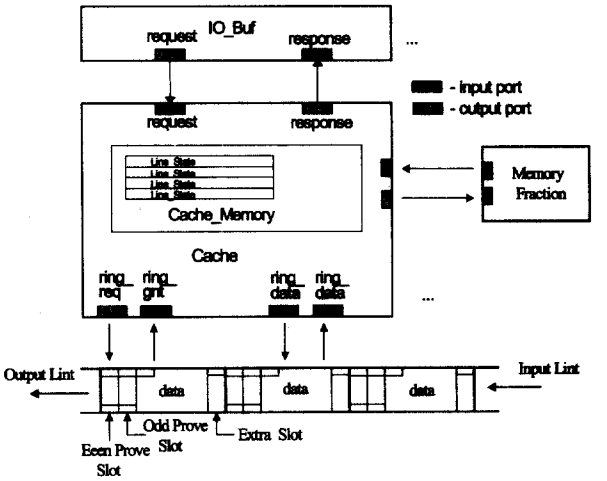
### 5.2 모의실험 환경

본 논문에서는 두 번째 성능평가 방법으로써 실행구동(Execution-driven) 방식의 모의실험기인 Limes[8]를 사용하여 모의실험을 수행하였다. Limes는 i486이상의 LINUX를 탑재한 PC에서 수행되며 크게 커널(Kernel) 부분과 모의실험기 부분으로 나뉘어진다. 커널은 응용프로그램과 모의실험기의 중간 부분에 위치하게 되며 스레드 기반(Thread-based)의 응용프로그램을 입력으로 받아서 수행한다. 이때에 메모리 참조 등의 사건(Event)이 발생하면 모의실험기 모듈을 호출하여 사건을 넘겨주게 된다. (그림 6)는 모의실험 환경 구성을 나타낸 것이다.




(그림 6) 모의실험 환경 구성

모의실험기 모듈은 입력점(Input Port)과 출력점(Output Port)을 통하여 시스템내 구성 요소들과 통신을 하게 되며 시스템의 상호 연결망과 메모리 계층 구조를 모델링한다. 또한 매 사이클마다 커널이 전송한 사건들을 분석하여 적절한 연산을 수행한 다음 커널이 다시 수행할 수 있도록 그 결과를 통보한다. Limes는 병렬 스레드들을 직접 스케줄링(Scheduling)하며 각 스레드에 관련된 정보를 명령 단위로 추적하기 때문에 실제 시스템에서 실행하는 것과 동일한 효과를 얻을 수 있다. (그림 7)은 시스템 모의실험기 모듈을 세부적으로 나타낸 것이다.




(그림 7) 모의실험기 내부 구성

### 5.3 모의실험 변수

모의실험은 2장에서 기술한 시스템 구조와 3장의 프로토

콜을 사용한다. 모의실험에 사용되는 시스템 구성 변수는 <표 2>와 같다. 목표 시스템은 중·소형 규모의 다중처리기 시스템을 가정하여 처리기 수를 16, 32, 64개인 경우에 대하여 실험하였다. 처리기수에 따라 링의 크기가 변화하며 링을 순환하는 프레임의 수가 결정된다. 분산 공유메모리 다중처리기 시스템에서는 분산된 메모리 모듈에 대한 참조 비율에 따라 성능에 영향을 준다. 본 논문에서는 각 처리기가 활성화된 후 참조의 50%를 지역 메모리 모듈에서 참조하도록 하였고 나머지는 각각의 모듈에서 무작위로 참조하도록 하였다.

〈표 2〉 시스템 구성 변수

| | |
|---|---|
| 처리기 수 | 16, 32, 64 |
| 캐쉬 사상(mapping) | 2-방향 집합 연관 (2-way set-associative) |
| 캐쉬 크기 | 8 Kbyte |
| 캐쉬 블록 크기 | 32 byte |
| 캐쉬 블록 수 | 256 개 |
| 메모리 참조 | 20 사이클 |
| 캐쉬 참조 | 1 사이클 |

각 처리기는 요구한 메모리 참조가 만족될 때까지 대기 상태에 있으며 처리기의 요구가 캐쉬에서 만족되었을 경우를 1사이클로 간주하였다.

### 5.4 응용프로그램

본 논문에서 사용하는 응용프로그램은 스레드 기반의 SPLASH 2[16]에서 제공하는 프로그램중에서 fft, barnes, lu, water-spatial등 4개를 사용한다. fft는 여섯 단계의 FFT알고리즘을 수행하며, Lu는 밀집행렬을 하위 삼각행렬과 상위 삼각 행렬로 인수 분해하는 응용프로그램이다. 또한 water-spatial은 액체 상태에서 물분자의 힘과 전위를 측정한다. <표 3>은 모의실험에서 사용된 응용프로그램에 주어진 문제 크기와 명령인자이며 공유 메모리에 대한 참조 횟수와 크기는 명령 인자들에 의해 차이가 있다

〈표 3〉 응용프로그램 문제크기 및 명령인자

| 응용프로그램 | 문제크기 | 명령인자 |
|---|---|---|
| FFT | 1M complex doubles | -m12 p32 n65536 l4 |
| BARNES | 512 nbody | Input.32 |
| LU | 128×128 matrix | -n128 p32 b16 |
| WATER | 512 molecules | Input.32 |

※ 처리기 수가 32개일 경우

<표 4>는 모의실험에서 사용된 응용프로그램이 수행되는 동안 요구한 메모리 참조 형태이다.

〈표 4〉 응용프로그램 메모리 참조 특성

| 응용 프로그램 | 총 계 | 읽 기 | | 쓰 기 | | Lock | |
|---|---|---|---|---|---|---|---|
| | | 횟수 | 비율(%) | 횟수 | 비율(%) | 횟수 | 비율(%) |
| Fft | 202,110 | 124,876 | 61.79 | 76,287 | 37.75 | 474 | 0.23 |
| Barnes | 23,802,911 | 16,694,638 | 70.14 | 7100,843 | 29.83 | 3,718 | 0.02 |
| Lu | 4,943,361 | 3,474,947 | 70.30 | 1,465,955 | 29.66 | 1,230 | 0.02 |
| Water | 8,098,777 | 6,120,107 | 75.57 | 1,966,733 | 24.28 | 5,967 | 0.07 |

※ 처리기 수가 32개일 경우

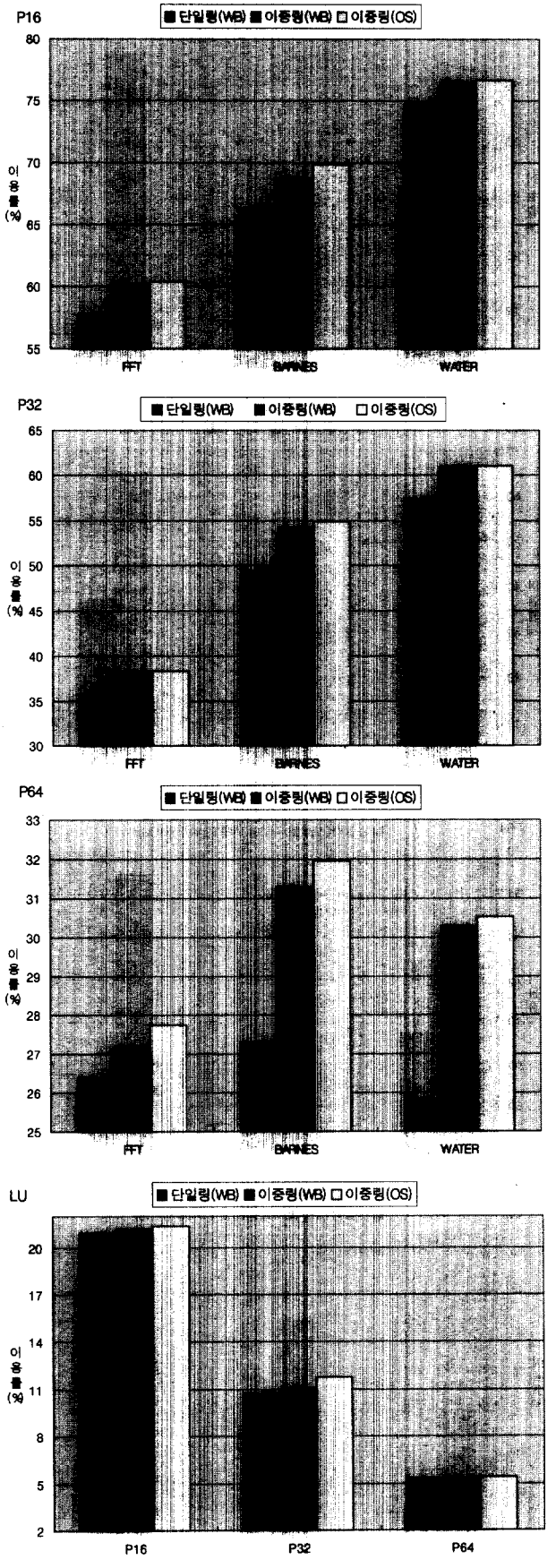
### 5.5 모의실험 결과

본 논문에서는 단일 슬롯링 구조와 이중 슬롯링에 대하여 지연쓰기와 기록 무효화 정책을 사용하는 프로토콜과 캐쉬가 블록소유(Ownership) 권한을 가지는 기록 무효화 정책을 적용하여 모의 실험을 하였다. 모의 실험 결과로서 성능에 영향을 미치는 중요 요소인 처리기 이용률, 접근지연시간, 링 이용률 등을 측정하여 양 시스템간에 성능을 비교하였다.

#### 5.5.1 처리기 이용률

처리기 이용률은 처리기 전체 실행 시간중 메모리 참조 대기 시간을 제외한 시간의 비율이며 시스템 성능에 영향을 미치는 중요 요소중의 하나이다. 처리기들은 응용프로그램 실행 기간중 메모리 참조 요구를 하게 되며 요구한 메모리 참조 요구가 만족될 때까지 대기 상태에 있게 된다. 메모리 참조 요구에 대한 접근지연 시간을 줄이는 방법중의 하나가 처리기와 메모리 사이에 고성능의 작고 빠른 캐쉬 메모리를 설치하는 것이다[1]. 이러한 캐쉬 메모리의 설치는 캐쉬 일관성 문제를 야기시키며 캐쉬 적중률을 향상시키는 데에는 한계가 있어 메모리 접근지연 시간을 줄이는 데에는 한계성을 지닌다.

또한 캐쉬에서 참조요구가 적중되더라도 쓰기 요구일 경우 공유 블록이면 메모리로부터 쓰기 권한을 얻어야 하며 해당 블록의 쓰기 요구가 다른 모든 캐쉬에 전파되어야 한다. 따라서 공유 메모리 다중처리기 시스템에서의 상호 연결망은 많은 처리기들을 수용할 수 있는 구조인 동시에 각 처리기들간에 통신 시간이 짧은 구조이어야 한다. (그림 8)의 결과는 처리기 수가 16, 32, 64개인 경우에 단일링과 이중링의 시스템에 대한 응용프로그램의 실행 결과이다. 그림에서 알 수 있듯이 모든 응용프로그램에 대하여 단일링보다는 이중링 구조에서 처리기 이용률이 향상되었음을 보여주고 있다. 또한 이중링 구조에서 지연쓰기보다는 캐쉬가 블록 소유권한을 유지하는 정책을 사용하는 프로토콜이 좋은 성능을 보여주고 있음을 알 수 있다.

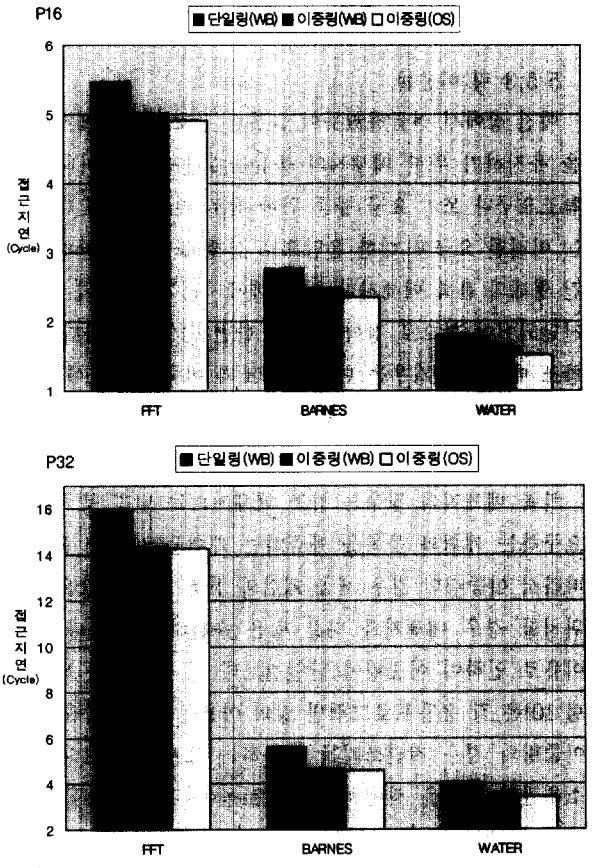


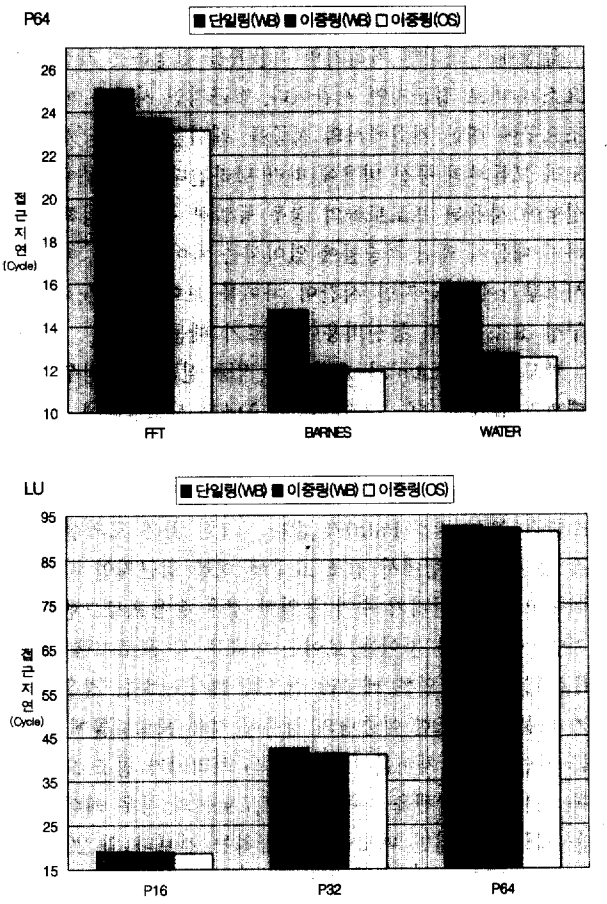

(그림 8) 응용프로그램별 처리기 이용률

### 5.5.2 응용프로그램별 접근지연 시간

앞장에서 논의한 처리기의 이용률을 결정하는 가장 중요한 요소가 바로 접근지연 시간이다. 접근지연 시간은 메모리 참조요구에 대한 캐쉬에서의 적중률, 캐쉬들간 또는 캐쉬와 메모리 모듈과의 통신 비용에 따라 달라진다. 본 논문에서 모의실험에 사용된 프로토콜의 경우 동일한 무효화 정책을 사용하기 때문에 캐쉬 적중률에 있어서는 차이가 없다. 따라서 양 시스템간에 접근지연 시간이 차이를 보이는 것은 시스템내 구성 요소들간에 통신비용이 다르기 때문이다.

메모리 참조 요구가 캐쉬에서 적중 실패했을 경우 만족될 때까지 필요로 하는 시간은 참조요구가 시스템내에 전파되는 시간과 요구 블록이 해당 캐쉬에 전달될 때까지 소요되는 시간이다. 이중링 구조는 단일링 구조보다 시스템내의 구성 요소들간 통신 경로(Path)가 짧다. (그림 9)은 모의실험 동안 각 응용프로그램에서 참조 요구에 대한 접근지연 시간을 보여주고 있다. 처리기 수가 동일할 경우 응용프로그램별 접근지연 시간이 큰 차이를 보이는 이유는 각 프로그램 별로 락(Lock) 등으로 인해 메모리 참조를 재시도 하는 경우가 다르기 때문이다. 또한 이중링의 경우에 적용 프로토콜에 따라 접근 지연시간이 차이를 보이는 것은 처리기가 참조 요구시 블록 소유권한 정책을 사용하는 프로토콜의 경우 캐쉬가 블록을 제공하게 되면 메모리가 제공하는 것보다 약 20배 정도 빠르게 응답할 수 있기 때문이다.



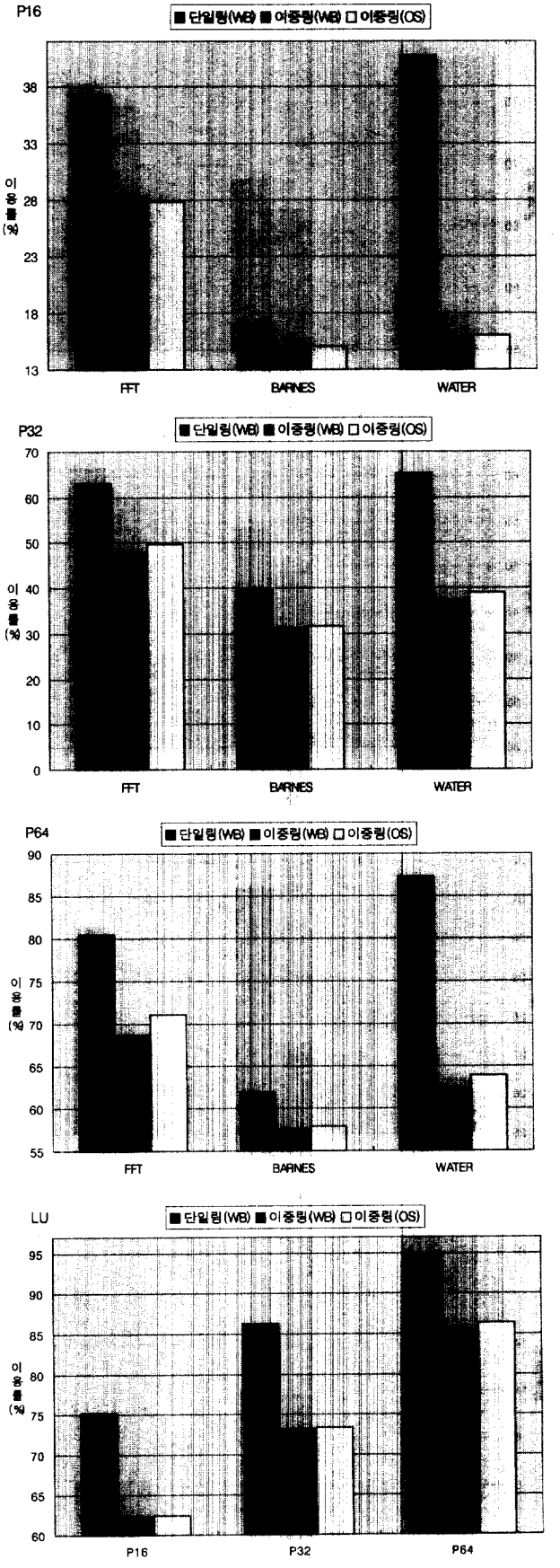



(그림 9) 응용프로그램별 접근지연 시간

5.5.3 링 이용률

이전 장에서 언급하였듯이 슬롯링 상호 연결망에서 일관성을 유지하기 위한 메세지의 종류는 2가지이다. 하나는 블록 참조요구나 쓰기 요구 등을 위한 참조 메세지이며 다른 하나는 이러한 요구에 대한 응답으로 실제 블록을 포함하고 있는 긴 형태의 블록 메세지다. 참조 메세지는 모든 처리기 캐쉬에 전달되어야 함으로 정확히 링을 한번 순환해야 한다. 따라서 참조 메세지의 경우는 단일링 구조와 이중링 구조에서 링 순환에 큰 차이가 없다. 그러나 블록 메세지의 경우는 조금 다르다.

블록 메세지의 경우는 참조 메세지와는 달리 블록을 참조 요구하는 캐쉬와 블록을 제공하는 캐쉬들 사이만 순환한다. 따라서 이중링의 경우 제공 캐쉬와 참조 캐쉬가 같은 링상에 위치할 경우 메세지가 다른 링은 순환하지 않는다. 이러한 이유로 인하여 이중링이 단일링보다 링 이용률이 낮다. (그림 10)은 각 응용프로그램별 링의 이용률을 보여주고 있으며 이중링의 경우 지연쓰기를 사용하는 프로토콜 보다는 블록 소유권한 정책을 사용하는 프로토콜이 링의 이용률이 같거나 조금 높음을 보여준다.




(그림 10) 응용프로그램별 링 이용률

이는 처리기가 참조 요구시 블록 소유권한 정책을 사용하는 프로토콜의 경우 캐쉬가 블록을 빠르게 제공하게 되며 지연쓰기를 사용하는 프로토콜 보다 상대적으로 빈 슬롯으로 존재할 수 있는 시간이 짧기 때문이다.

## 6. 결 론

본 논문에서는 공유버스의 대안으로 제시된 점대점 상호연결망인 단일 슬롯링의 경우 시스템의 처리량을 높이기 위하여 처리기의 수를 증가시킬 경우 처리기들을 연결하는 링 크기의 증가로 인해 메모리 접근지연 시간이 증가하여 성능이 감소하는 단점을 보완하기 위한 이중 슬롯링 구조를 제안하였다. 또한 기존의 지연쓰기 정책을 사용하는 프로토콜보다는 캐쉬의 블록 소유권한을 유지하는 정책을 사용하는 새로운 스누핑 일관성 프로토콜을 사용함으로써 메모리 모듈에 대한 통신량을 줄였다. 이는 메모리 참조 요구에 대하여 메모리 보다 매우 빠르게 동작하는 캐쉬가 응답을 할 수 있게 함으로써 블록 참조 요구에 대한 응답시간을 줄였다.

새로 제안된 이중 슬롯링 구조는 처리기가 16, 32, 64인 중형 시스템에서 다양한 응용프로그램 환경하에서 기존의 단일 슬롯링 구조보다 좋은 성능을 얻을 수 있음을 모의실험을 통하여 확인하였다.

## 참 고 문 헌


[1] Per Stenstrom, "A Survey of Cache Coherence Schemes for Multiprocessor," IEEE Computer, pp.12-24, Jun. 1990.

[2] L. A. Barroso and M. Dubois, "Cache Coherence on a Slotted Ring," Intl. Conf. on Parallel Processing, pp.I230-I237, 1991.

[3] L. A. Barroso and M. Dubois, "The Performance of Cache-Coherent Ring-based Multiprossors," Proc. 20th Annul. Intl. Symp. on Computer Architecture, pp.268-277, May, 1993.

[4] M. Dubois, "Cache Architectures in Tightly Coupled Multiprocessors," IEEE Computer, pp.9-11, June, 1990.

[5] M. Dubois and F. Briggs, "Effect of Cache Coherency in Multiprocessors," IEEE Tran. on Comuter, No.11, pp.1083-1099, Nov. 1982.

[6] J. Archibald and J. L. Bear, "Cache Coherence Protocols : Evaluation Using a Multiprocessor Simulation Model," Acm. Trans. Comput. Sys. Vol.4, pp.273-298, Nov. 1986.

[7] D. Chaiken et al, "Directory-Based Cache Coherence in Large-Scale Multiprocessor," IEEE Computer, pp.49-57, June, 1990.

[8] Davor Magdic. Limes : A Multiprocessor simulation Environment for PC Platforms(http : //galeb.etf.bg.ac.yu/~dav0r/limes).

[9] Z. Vranesic, M. Stumm, D. Lewis and R. White, "Hector : A hierarchically Structured Shared Memory Multiprocessor," IEEE Computer, Vol.24, No.1, pp.72-78, January, 1991.

[10] Kendall Square Research, "Technical Summary," Walthan, Massachusetts, 1992.

[11] D. Gustavson, "The Scalable Coherence Interface and Related Standards Projects," IEEE Micro, Vol.12, No.1, Februrary 1992.

[12] L. Censier, and P. Feautrier, "A New Solution to Coherence Problems in Multicache Systems," IEEE Trans. On Computers C-27(12), pp.1112-1118, December, 1978.

[13] D. Chaiken, C. Fields, K. Kurihara and A. Agawal, "Directory-Based Cache Coherence in Large Scale Multiprosessors" IEEE Computer, Vol.23, No.6, pp.49-59, June, 1990.

[14] L. Lamport, "How to Make a Multiprocessor Computer that correctly executes Multiprocess Programs," IEEE Trans. on Computers, Vol.C-28, No.9, pp.690-691, Sept. 1979.

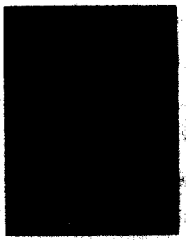
[15] C. Scheurich and M. Dubois, "Correct Memory Operation of Cache-based Multiprocessors," The 14th Intl. Symp. on Computer Architecture, pp.234-243, 1987.

[16] S. WOO, and J. Singh. The SPLASH2 Programs : Characterization and Methodological Considerations. In Proceedings of the 22nd Annual Int'l Symp. On Computer Architecture, 43-63, June, 1995.

[17] PANDA : Ring-Based Multiprocessor System using New Snooping Protocol," Sung Woo Chung, Seong Tae Jhang, Chu Shik Jhon, ICPADS'98(International Conference on Parallel And Distributed Systems), pp.10-17, December, 1998.

[18] "STARRING : Slotted Ring_Based Multiprocessor System with a Central Directory Module," Seong Tae Jhang, et al., 1997 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, Canada, Aug. 20-22, 1997.

[19] A. Grbic, S. Brown, S. Caranci, R. Grindley, M. Gusat, G. Lemieux, K. Loveless, N. Manjikian, S. Srbljic, M. Stumm, Z. Vranesic, and Z. Zilic, "Design and Implementation of the NUMAchine Multiprocessor," To appear in Proceedings of the 35th IEEE Design Automation Conference, San Francisco, June, 1998.




## 민 준 식

e-mail : jsmin@nca.or.kr

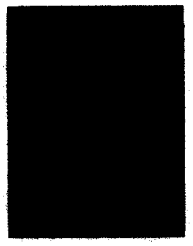
1994년 동국대학교 컴퓨터공학과 석사과정 졸업(공학석사)

2001년 동국대학교 컴퓨터공학과 박사과정 졸업(공학박사)

1994년~1996년 쌍용정보통신 연구원

1997년~현재 한국전산원 국가망 이용관리 팀장

관심분야 : 컴퓨터 구조, 병렬처리, 망관리 등


## 장 태 무

e-mail : jtm@dgu.ac.kr

1977년 서울대학교 전자공학과 졸업(학사)

1979년 한국과학기술원 전산학과 졸업 (이학석사)

1995년 서울대학교 컴퓨터공학과 졸업 (공학박사)

1979년~1981년 한국전자기술연구소 연구원

1998년 University of Southeastern Lousiana 교환교수

1981년~현재 동국대학교 컴퓨터·멀티미디어공학과 교수

관심분야 : 분산 및 병렬처리, 컴퓨터 구조, 입출력시스템 등