論 文

다중 버스 시스템에서 감소된 제어라인을 갖는 중재기의 설계

崔 昌 勳*, 韓 昌 熙**, 金 聖 大*

A Design of Arbiter with Reduced Control Lines In Multiple Bus System

Choi Chang Hoon*, Han Chang Hee**, Kim Sung Chun* Regurlar Members.

要 約

다중 버스 상호연결망은 단일 버스의 구조를 적용하여 쉽게 확장할 수 있고, 다른 상호연결망보다 비교적 신뢰도가 높은 장점을 갖고 있다. 그러나 중재기의 구조와 중재 기법이 복잡하고, 중재에 사용되는 제어라인이 매우 많이 필요하므로 구현이 용이하지 못하다.

본 논문에서 제안하는 중재기와 중재 기법은 기존의 두 단계 중재를 한 단계로 줄이고, 제어라인의 수를 대폭 감소시킨 구조이다. 또한 적절한 메모리 모듈 수를 결정하여 기존의 중재에 걸리는 시간과 비슷한 중재 시간을 갖게 함으로써 성능의 저하를 방지할 수 있다.

본 논문은 위와 같은 특성을 갖는 파이프라인화된 병렬 중재기의 구조와 중재 프로토콜을 제시하였으며, 시뮬레이션을 이용한 성능 분석을 통하여 제안된 중재기를 사용하는 다중 버스 시스템의 성능적 특성과 최적의 비용으로써 시스템을 구성 할 수 있는 방법을 제시하였다.

ABSTRACT

The multiple bus interconnection network not only can be easily extended with a single bus structure, but also has the advantage of being one of the most reliable structures in interconnection networks. However the exsiting arbiter structures and arbitration schemes are inefficient when implemented due to the fact that the arbitration scheme is complex and requires too many control lines for arbitration.

Therefore, the proposed arbiter and arbitration scheme in this paper, reduces two-stage arbitration to a single-stage and also makes the number of contol lines significantly reduced. And since we determine a proper number of memory modules, the proposed scheme provides an arbitration time

* 西江大學校 電子工學科
Dept. of Computer Science, Sogang University.
** 삼성데이타 시스템 품질관리팀
Samsung Data Systems Co., LTD. Quality Management Team

論文番號 : 9476
接受日字 : 1994年 3月 10日

similar to the exsiting arbitration time. So that performance degradation of the system can be prevented.

In this paper, we show a pipelined parallel arbiter and an arbitration protocol which has the above properties. And we characterize a multiple bus system with the proposed arbiter through simulation for performance analysis and propose methodology for organizing an optimal cost system.

I. 서 론

공유 메모리(shared memory)를 가지는 밀결합(tightly coupled) MIMD 다중프로세서(multiprocessor)컴퓨터 시스템은 프로세서와 공유메모리를 연결하는 방법에 따라서 시스템의 설계 특성 및 성능이 크게 좌우되게 된다. 프로세서와 공유메모리 간의 연결을 상호연결망(interconnection network)이라 하며, 크게 단일버스(single bus)상호연결망, 크로스바(cross-bar) 상호연결망, 다단계 상호연결망(MIN:Multistage Interconnection Network), 그리고 다중버스(multiple bus)상호연결망으로 나눌 수 있다. 특히 단일 버스 구조는 다중버스에서 버스의 수가 하나인 경우로 볼수 있다 [1,2].

다중 버스 구조는 프로세서 모듈, 메모리 모듈 그리고 버스 수의 조합에 따라 성능 및 비용에 많은 차이가 있지만 단일 버스와 크로스바의 중간 정도에 위치한다. <그림1>은 N개의 프로세서 모듈, M개의 메모리 모듈, B개의 버스를 갖는 N×M×B 다중 버스 구조이다. 이 구조는 그 특성상 높은 신뢰도와 단일 버스에서 적용되는 간단한 구조를 쉽게 확장, 적용 가능하다는 이점을 가지고 있다 [3-8].

다중 버스 구조에서는 프로세서의 수에 비하여 한정된 버스와 공유하는 메모리 모듈에서의 충돌을 해결하기 위하여 중재기(arbiter)가 필요하다. 중재기의 특성은 다중버스의 구조에 따라서 차이가 있다. 일반적으로 다중버스는 버스 동작의 동기화 여부에 따라서 동기식(synchronous)과 비동기식(asynchronous), 교환 방식에 따라서 회선 교환(circuit switchint)과 패킷 교환(packet switching), 그리고 중재기 또는 버스 제어기의 위치에 따라서 중앙 집중형(centralized)과 비집중형(decentralized) 또는 분산형(distributed)으로 나눌 수 있으며, 이들의 각 조합에 따라서 8가지의 다중 버스로 구분할 수 있다[8]. 중재기의 구조 및 특성과 구성도 위 3가지의 분류 방법에 따라서 달라지게 되고, 이를 설계에 반영해야 한다.

그리고 중재기의 설계시 고려해야할 사항으로는 충돌에 의한 경쟁시에 중재 순서를 결정하기 위한 우선순위를 선정하는 방법과 다중 버스 구조에서만 독특하게 나타나는 중재 특성으로 메모리에 대한 점유와 버스의 점유 순서에 따라서도 중재기의 구조가 변경된다는 사실이다[2,9-16].

다중 버스에서의 중재는 두 단계로 이루어진다. 일반적으로는 우선 메모리 모듈을 점유하고 후에 버스를 점유한다. 이러한 방법이 버스를 먼저 점유하는 방법보다 효율적이다. 따라서 메모리를 점유하는 과정에서의 충돌과 버스를 점유하는 과정에서의 충돌을 해결하기 위한 두 가지 종류의 중재기가 필요하다[2].

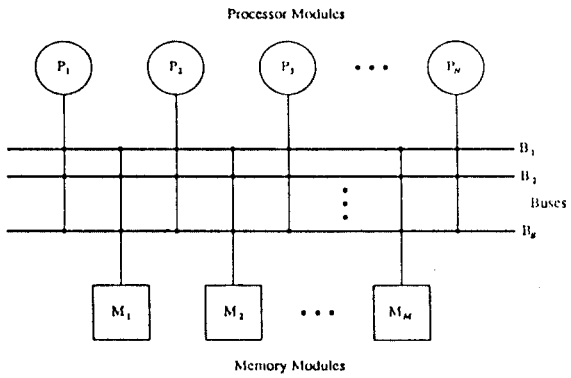


그림 1. N×M×B 다중 버스 시스템

다중 버스 구조에서 나타나는 문제점 중의 하나는 중재기의 구조가 복잡하며, 특히 중재를 위한 제어라인(control line)의 수가 매우 많이 필요하다는 것이다. 따라서 다중 버스의 크기가 커지게 됨에 따라서 중재에 걸리는 오버헤드가 증가하고, 제어라인의 수가 많아지게 되므로 대형 시스템에 적용하기 어렵다 [10].

본 논문에서 제안하는 중재기의 구조와 중재 방법 및 프로토콜은 위에서 지적한 문제점을 해결하기 위한

방안으로 중재 과정을 한 단계로 줄였으며, 중재기를 위한 제어라인의 수를 대폭 축소하였다.

본 논문의 구성 및 내용은 다음과 같다. 2장은 단일 버스인 IEEE Future Bus에 적용된 경쟁 논리 회로를 다중 버스에 적용하여 중재기의 구조와 중재 방법 및 프로토콜을 제시하였다. 3장에서는 2장에서 제시한 중재기의 성능을 평가하였고, 일반적으로 적용하고 있는 다중 버스 중재기와 성능 및 특성을 비교하였다. 4장에서 본 논문의 결론을 맺는다.

Ⅱ. 파이프라인화된 병렬 중재기

본 논문에서 제안하는 파이프라인화된 병렬 중재기(pipelined parallel arbiter)는 두단계의 중재과정을 한 단계로 줄이고, 또한 제어라인의 수를 대폭 감소시킨 구조이다. 중재 과정을 한 단계로 줄였지만 1-of-N중재기[12]와 비슷한 중재 지연시간을 갖고 있으며, 제어라인의 수는 약 $\lceil \log_2 N \rceil$정도의 기본 제어라인을 갖게 된다. 또한 완전한 분산 중재기 (distributed arbiter)로 고장에 강한 (fault-tolerence)특성을 갖는다. 이러한 파이프라인화된 병렬 중재기의 세부적인 구조와 중재 방법 및 프로토콜은 다음과 같다. 이때 본 논문에서 제시하는 중재기 구조와 프로토콜은 일반적으로 고려되는 회선 교환 분산형 비동기식 다중 버스 구조에 적용하기로 한다.

2.1 경쟁 논리 회로

중재기의 주요한 역할인 충돌 해결을 위한 기본적인 경쟁 논리 회로(competition logic circuit)는 〈그림 2〉에 나타나 있다. 이회로는 다중 프로세서를 위한 단일버스인 IEEE Future버스에서 채택되어 표준화가 되어있다[14,15].

이 경쟁 논리 회로의 동작은 COMPETE신호가 HI일때, 각 프로세서에서 경쟁이 시작되는데 각 프로세서에 중재 번호(arbitration number)가 배정되어 있어 이것이 우선 순위의 역할을 한다. 중재 번호가 높을수록 우선순위가 높다. 프로세서에서 메모리의 사용 요구가 발생하면 프로세서가 갖고 있는 각각의 중재 번호를 wired-OR되어 있는 공용의 중재라인(arbitration line)에 올리게 된다. 중재라인에 올라와 있는 번호는 경쟁에 돌입한 각 프로세서의 중재 번호가 모두 OR되어 있는 형태로 이 번호를 최상위 비트(most significant bit)부터 최하위 비트(least significant bit)까지 차례로 비교한다.

비교 도중에 한 비트라도 자신보다 큰 수를 만나게되면 경쟁에서 패배하게 되어 즉시 자신의 중재 번호를 중재라인으로 부터 회수한다. 최하위 비트까지 모두 자신의 번호보다 낮거나 같은 수와 비교된 경우 이때 중재라인에 남아 있는 번호는 자신의 중재 번호와 동일하게 되고 경쟁에서 승리하여 버스의 사용권을 획득한다.

이와같은 경쟁회로는 단일 버스에서는 버스가 한개인 관계로 쉽게 적용하여 이용할 수 있지만 다중 버스에서는 추가적인 프로토콜을 첨가해야 한다. 즉 다중 버스에서는 이용 가능한 버스의 수가 한개이상일 수 있으므로 한 단계의 경쟁이 끝난 뒤에도 사용 가능한 버스가 존재한다면 즉시 재경쟁에 돌입해야 하기 때문이다.

2.2 버스 동작

2.2.1 시스템 및 중재기의 구성

위에서 제시한 경쟁 논리회로를 갖는 중재기를 사용하는 다중 버스 시스템의 구조는 〈그림3〉에 나타나 있다. 〈그림3〉에서 프로세서 모듈과 메모리 모듈은 버스 중재시 기본적으로 필요한 장치만을 간략하게 도식화하였다. 모든 프로세서 모듈에 분산된 중재기에는 〈그림2〉과 같은 경쟁 회로와 버스와 상태가 저장된 버스상태 플래그(bus status flag) 그리고 중재 번호가 저장된 레지스터 등으로 구성된다.

중재를 위한 메모리 모듈의 장치로는 어드레스 디코더(address decoder)와 메모리의 상태가 기록되는 메모리 상태 플래그(memory status flag)등이 각 메모리 모듈에 존재한다. 버스는 크게 중재 버스(arbitration bus)와 데이타와 어드레스 버스(data and address bus)로 구성된다. 중재 버스는 프로세서에 의해 버스와 메모리 모듈이 점유되어 실제 필요한 데이타가 전송되기 전까지의 모든 동작을 제어하고, 데이타 전송이 완료되어 점유한 메모리와 버스를 해제하는데 필요한 모든 동작 신호들의 전송로로 이용된다. 데이타와 어드레스 버스는 중재 동작이 완료

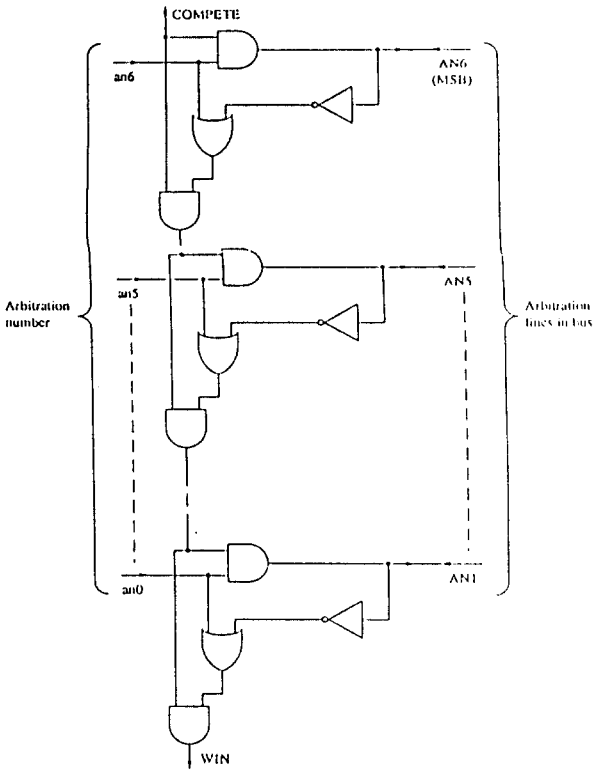


그림 2. 경쟁 논리 회로

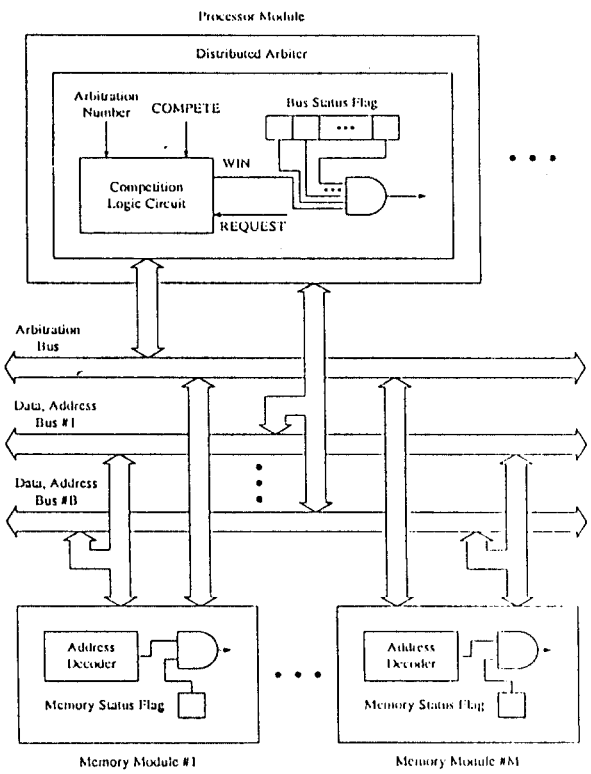


그림 3. 파이프라인화된 방렬 중재와 시스템 구성도

되어 프로세서와 메모리간의 데이타의 전송에 필요한 전송로로써 데이타 전송에 필요한 제어 신호를 위한 제어 버스 (control bus)도 여기에 포함된다.

본 시스템은 회선 교환 버스이므로 메모리와 프로세서간의 전송로, 즉 버스가 설정이 된다면, 이미 단일 버스에서 이용되는 데이타 전송 프로토콜을 그대로 이용할 수 있다. 따라서 본 논문에서는 데이타 전송에 관련된 프로토콜의 서술은 생략하기로 하고, 데이타가 전송되기전 버스 중재 및 제어와 데이타 전송후에 버스해제에 관련된 중재와 제어동작 및 프로토콜만을 서술하기로 한다. 중재 버스는 중재에 관련된 모든 제어 신호와 상태 신호의 전송로 역할을 하는 버스로 프로세서, 메모리, 버스의 상태와 중재 버스는 다음과 같이 구성된다.

Processor Status(PS) : PROCESSING
REQUEST
CANDIDATE
MASTER

Memory Status(MS) : IDLE
BUST

Bus Status (BS) : IDLE
BUSY

Arbitration Bus : AL(Arbitration Line)
CE(Competition Enable)
ME(Memory Status Check Enable)
MA(Memory Module Address)
MS(Memory Status)
BS(Bus Status)

Processor Status는 프로세서와 메모리 간의 데이타 전송과정에서 버스와 메모리 획득 절차에 따른 상태를 나타내는데, 프로세서의 상태는 PROCESSING, REQUEST, CANDIDATE, MASTER 네가지로 나룰 수가 있다. 이러한 각 프로세서의 상태들에 대한 설명은 3.2.2 버스 동작 및 프로토콜 부분에서 상세히 설명하기로 한다. Memoruy Status는 메모리의

상태로서 임의의 프로세서에 의해 메모리가 사용중인 경우는 BUST상태이고, 메모리를 사용하는 프로세서가 없는 경우 IDLE상태이다. 버스의 상태 역시 프로세서의 점유 여부에 따라서 Bus Status는 IDLE과 BUSY상태로 구분할 수 있다. 메모리의 상태와 버스의 상태 신호는 이를 점유하는 프로세서에 의해 변경된다.

중재 버스는 여섯 종류의 라인으로 구성되는데, AL은 중재를 위한 라인으로 각 프로세서의 경쟁 회로에서 출력되는 중재 번호와 wired-OR된 형태이다. CE는 메모리를 요구한 프로세서들이 경쟁에 돌입하게 하는 제어라인이다. ME는 경쟁에서 승리한 프로세서가 메모리 모듈 상태의 확인 가능하게 하는 제어라인을 나타낸다. 이 두가지 CE와 ME에 실리는 신호는 항상 경쟁의 승자를 한 프로세서로 유지하기 위한 동기 신호이다. MA는 경쟁에서 승리한 프로세서가 메모리의 상태를 확인하기 위하여 MA라인을 통하여 확인하고자 하는 메모리 모듈의 주소를 전송한다. MS는 MA라인에 실려있는 메모리 모듈의 주소에 해당하는 메모리의 상태신호를 전송하는 라인이다.

2.2.2 버스 동작 및 프로토콜

2.2.1과 같이 정의 된 시스템에서 프로세서와 메모리간의 데이타 전송 동작은 다음과 같다.

각 프로세서에는 전용 캐쉬(private cache)가 존재하는데 프로세서에서 수행중인 명령어(instruction)와 데이타는 전용 캐쉬를 통해서 액세스하게 된다. 이때 전용 캐쉬에 명령어나 데이타가 준비되어 있지 않다면 공유 메모리로부터 전송을 해야 한다. 따라서 이러한 공유 메모리에서 데이타 전송을 요구하는 상태를 프로세서의 REQUEST상태라 정의한다. 프로세서가 REQUEST상태가 되면 중재 버스의 CE신호를 확인하고 CE가 HI 즉, 경쟁이 가능한 상태이라면 즉시 경쟁이 시작된다. 이러한 경쟁 방법은 경쟁 논리와 함께 위에서 설명이 되었기 때문에 여기서는 생략하기로 한다. 경쟁에서 승리한 프로세서는 CE에 LO신호를 활성화하고, IDLE한 상태의 버스가 발생할 때까지 대기한다. CE를 LO로 하는 이유는 경쟁에서 승리한 프로세서의 갯수를 한개로 하여 메모리에서의 충돌을 해결하려는 목적이다. 따라서 위와같은 방법에서는 메모리에서의 충돌이 발생하지 않으므로 메모리 중재가 필요없기 때문에 메모리에 중재기를 설치하지 않아도 된다.

IDLE한 상태의 버스가 존재하고 ME가 HI가 되면 경쟁에서 승리한 프로세서의 상태는 CANDIDATE가 된다. CANDIDATE상태인 프로세서는 다른 프로세서들이 경쟁이 가능하도록 CE를 HI로 전환하고, MA를 통하여 원하는 메모리 모듈의 어드레스를 전송하여 메모리의 상태를 확인한다. 메모리 모듈에는 메모리 모듈 어드레스 디코더가 있어서 만약에 자신의 어드레스가 MA를 통해서 전송되어 왔다면 이것을 감지하여 자신의 상태를 MS에 전송한다. MS를 통하여 메모리의 상태를 확인한 프로세서는, 만약 메모리의 상태가 IDLE이라면, 버스와 메모리를 점유하고 점유한 메모리와 버스의 상태를 BUSY상태로 전환시킨다. 버스를 BUSY상태로 전환할 경우는 BS에서 점유한 버스의 번호에 해당하는 라인에 HI신호를 활성화하고, 메모리는 메모리 상태 프래그에 1을 셋팅한다.

만약 메모리의 상태가 IDLE이라면 프로세서의 상태는 MASTER가 되고 데이타 전송을 시작한다. 만약 메모리의 상태가 BUSY라면 IDLE한 상태의 버스를 다른 프로세서가 이용할 수 있도록 CANDIDATE 상태에서 REQUEST 상태로 전환하고 경쟁이 가능하면 재경쟁에 들어간다.

데이타의 전송이 완료되면 점유한 메모리와 버스를 해제시키고 프로세서는 PROCESSING상태로 전환한다. 이러한 버스 동작에 따르는 프로세서의 상태 전이는 <그림4>로 대략적으로 요약할 수 있다.

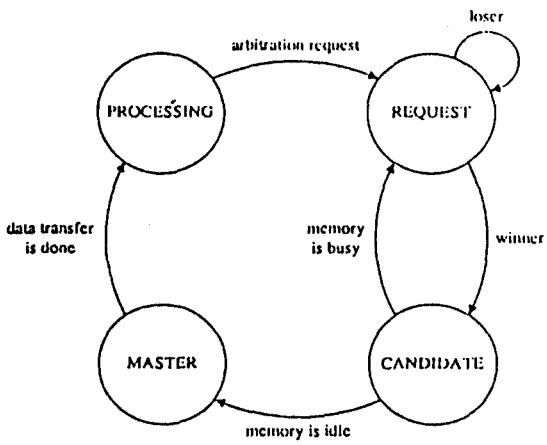


그림 4. 버스 동작에 따른 프로세서 상태 전이도

지금까지 서술한 버스 동작 프로토콜을 정형화하여 표현하여 〈그림5〉와 같은 흐름도(flow diagram)로 나타낼 수 있다.

위에서 서술한 버스 제어는 각 프로세서에 배치된 중재기에서 병렬적으로 이루어지며, 경쟁에서의 승자, 즉 후보는 파이프라인 형태로 연속적으로 발생하게 된다. 위에서 제시한 프로토콜은 고정 우선 순위 배정 방법으로 앞장에서 서술하였듯이 우선 순위 배정에 따라 성능에 많은 차이가 있다. 파이프라인화된 병렬 중재는 여러 우선 순위 배정 방법을 간단한 프로토콜 상의 추가로 구현할 수가 있는데 〈그림2〉와 같은 경쟁 논리 회로에서 중재 번호를 조정함으로써 이를 구현할 수 있다. 중재 번호를 특정 시점에서 다운 카운트 시키게되면 쉽게 순환 우선 순위 방법이 구현된다. 이러한 순환 우선 순위 방법을 위 프로토콜에 추가적으로 적용시켜 쉽게 확장이 가능하다.

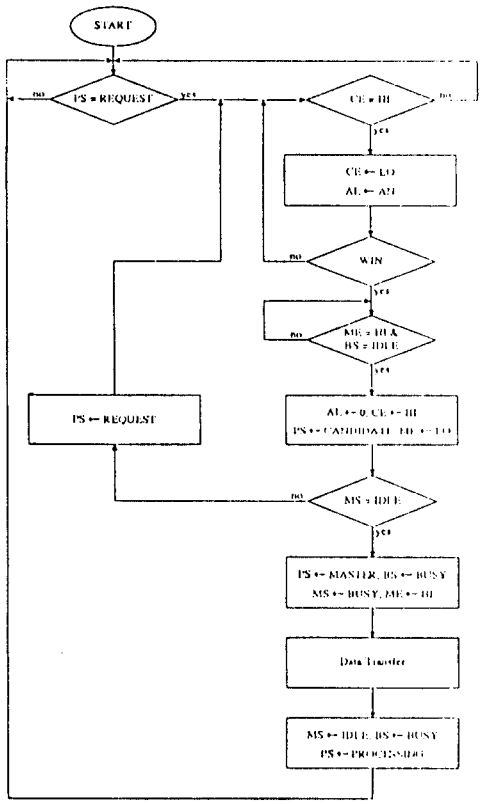


그림 5. 고정 우선 순위 중재 프로토콜 흐름도

Ⅲ 성능 분석 및 비교

2장에서 서술한 중재 프로토콜과 중재기를 사용하는 시스템의 성능을 분석하기 위하여 시뮬레이션(simulation)을 이용하였다. 본 장에서는 시뮬레이션을 위한 성능 매개 변수(performance parameter), 시뮬레이션 모델(simulation model), 시스템의 성능에 대하여 논한다.

3.1 성능 분석을 위한 다중 버스 모델

3.1.1 성능 매개 변수

성능 평가를 위한 성능 매개 변수는 다음과 같다.

N : number of processor modules
M : number of memory modules
B : number of buses
Ta : arbitration time at pipelined parallel arbiter
Tw : waiting time before holding memory module
Td : data transfer time
Tm : memory access time
Tave : average memory access time
ρ : request rate from processor
H : hit ratio at private cache
BVUSY : number of buses at busy status
MBUSY : number of memory modules at busy status
Pfail : probability of failure of holing memory

Ta는 각 프로세서에서 단한번의 경쟁에서 발생하는 지연 시간으로 $\log_2 N$배로 지연시간이 증가한다. Tw는 프로세서에서 메모리 사용을 요구한 뒤에 프로세서가 메모리를 점유할 때까지 걸리는 시간이다. 이 시간에는 중재 과정에서 걸리는 경쟁 시간, 경쟁에서 패배로 인한 재 경쟁시간과 메모리의 상태를 확인하는데 걸리는 시간들이 포함된다. Td는 프로세서가 메모리 모듈을 점유한 후에 메모리와 데이타를 전송하는데 걸리는 시간이다. Tm은 메모리 액세스 시간으로 메모리 모듈을 점유하는데 걸리는 시간(Tw)과 데이타를 전송하는데 걸리는 시간(Td)이 포함된다. Tave는 평균 메모리 액세스 시간으로 모든 프로세서의 총 메모리 액세스 시간을 총 요구횟수로 나눈 시간이다. 각 시간들 간의 관계는 다음과 같이 표현할 수 있다.

$Tm = Tw + Td$
$Tw < Ta$

프로세서가 수행중에 전용 캐쉬에 명령어나 데이타가 준비되지 않은 경우 즉, 캐쉬에서 실패(miss)가 발생하는 경우 공유 메모리로부터 데이타를 전송해와야 한다. 이러한 경우 메모리를 요구하는데 임의의 시점에서 요구율(request rate, ρ)의 확률로 요구를 발생한다. H가 캐쉬에서의 적중률(hit ratio)일때, 캐쉬의 실패율(miss ratio)는 1-H가 된다. 따라서 다음과 같이 요구율의 범위를 결정할 수 있다.

$\rho \leq (1 - H)$

2장에서 제안한 프로토콜은 개념적으로 버스 선점 방식에 해당한다. 따라서 중재 경재에서 승리한 프로세서가 메모리를 점유하기 위해서 메모리의 상태를 확인해야 하는데 메모리가 이미 다른 프로세서에 의

$$P_{fail} = \frac{M_{BUSY}}{M}$$

해서 점유되어 있다면, 메모리 액세스는 실패한다. 따라서 이러한 동작의 발생은 시스템의 성능 저하 요인이 되는데 중재 경쟁에서 승리한 프로세서가 메모리 점유를 실패할 확률은 다음과 같이 유도된다.
이때 BUSY한 상태의 메모리 모듈의 수는 BUSY한 상태의 버스의 수와 동일하므로 M_{BUSY}와 B_{BUSY}는 동일하다. 그리고 버스의 트래픽이 매우 많아서 한개의 버스

$$P_{fail} = \frac{B_{BUSY}}{M} \leq \frac{B - 1}{M}$$

라도 IDLE상태가 되면 중재 경쟁에서 승리하여 대기중인 프로세서에 의해 바로 메모리의 상태가 확인된다. 따라서 이러한 상황의 메모리 점유 실패율은 다음과 같이 유도할 수 있다.

3.1.2 시뮬레이션 모델

시뮬레이션을 이용한 성능 평가에 있어서, 시뮬레이션 수행시 아래 7가지 가정을 적용하기로 한다.

1) 모든 프로세서는 동일한 요구률로 메모리 요구를 발생시킨다.
2) 각 프로세서 모듈은 적중률 H 가 95%이상인 캐쉬를 갖으므로 요구률의 범위는 $0 < \rho \leq (1-H)$ 이다.
3) 각 프로세서가 특정 메모리 모듈을 액세스할 확률은 1/M로 균일 분포(uniform distribution)를 이룬다.
4) 데이타 전송 시간(Td)은 읽기 동작과 쓰기 동작에 상관없이 항상 일정하다. 데이타 전송시간은 한 요구에 대한 데이타 전송시 전송되는 블럭의 크기에 비례하는데 이 블럭의 크기는 캐쉬의 한 블럭 크기와 동일하다. 본 시뮬레이션 모델에서는 이 블럭의 크기를 정의하지 않고, 단지 데이타 전송 시간을 100단위 시간(unit time)으로 고정하였다.
5) 중재 시간은 프로세서의 로그수로 증가하는데 프로세서의 수가 2인 경우를 1단위 시간으로 설정하면 중재시간은 $\lceil \log_2 N \rceil$단위 시간을 갖는다.
6) 메모리 상태를 확인하는데 걸리는 시간을 1 단위 시간으로 한다. 데이타 전송시간, 중재 시간, 대기 시간, 메모리 상태확인 시간 이외의 다른 동작에 소용되는 지연은 없는 것으로 가정한다.
7) 버스의 배정은 버스의 번호가 작은 것부터 순차적으로 배정하였다. 따라서 버스 이용률(bus utilization)은 버스의 번호가 작은 것이 가장 높다.

시뮬레이션 모델은 위 7가지의 가정하에서, 3장에서 기술한 프로토콜을 그대로 반영하여 버스 동작을 수행한다.

3.2 성능 분석 결과 및 평가

성능 분석을 위해 기본적으로 64×64×8 다중 버스 시스템의 시뮬레이션을 수행하였다. 그리고 가변적인 입력 매개변수는 요구율, 프로세서의 수, 메모리 모듈의 수, 버스의 수와 중재방법으로 고정 우선 순위와 순환 우선순위를 이용하였다. 성능 측정을 위한 매개변수는 평균 메모리 액세스 시간과 버스 이용률로 한정하였다. 각 매개변수와 성능의 관계는 다음과 같다.
〈그림6〉은 요구율 증가에 따른 평균 메모리 액세스 시간의 변화를 나타내는데, 요구율이 증가함에 따라서 평균 메모리 액세스 시간이 증가하는 이유는 메모리를 요구한 프로세서의 수에 비하여 이용가능한 버스의 수가 한정되었기 때문에 버스가 이용가능할 때까지 대기하는 시간이 커지기 때문이다.

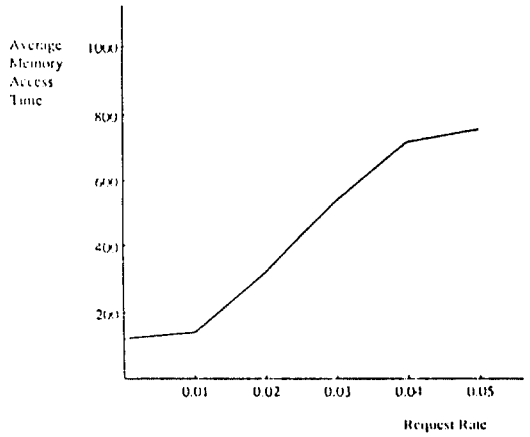
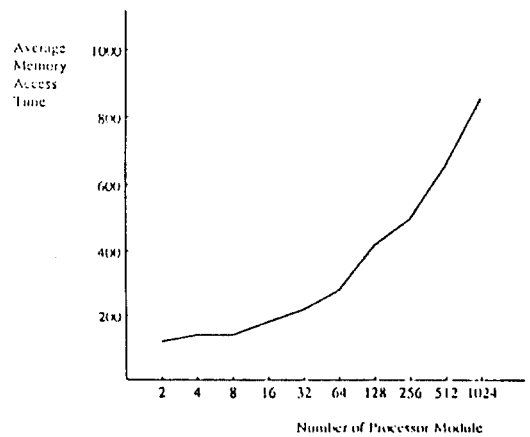


그림 6. 요구율 변화에 따른 평균 메모리 액세스 시간

고정 우선 순위와 순환 우선 순위 중재에 따른 평균 메모리 액세스 시간의 차이는 크지 않지만, 메모리 액세스 시간중에서 가장 길게 걸린 시간이 최악 메모리 액세스 시간이라 할 때, 이 두가지 방법의 버스 중재 기법을 적용한 시스템의 최악 메모리 액세스 시간의 비교는 〈표1〉 나타나 있다. 고정 우선 순위의 최악 메모리 액세스 시간이 요구율이 증가함에 따라서 크게 증가하는 이유는 우선순위가 낮은 프로세서에서 기아현상이 나타나기 때문이다. 〈그림7〉은 메모리 모듈의 수를 64개, 버스의 수를 8개로 고정하고 프로세서의 수를 2개에서 부터 1024까지 증가시킬때 평균 메모리 액세스 시간이다. 그래프에서 나타난 것처럼 프로세서의 수를 증가시키는 것은 프로세서의 수를 한정시키고 요구율을 증가시킨 결과와 동일한 의미를 갖는다.



〈그림7〉 프로세서의 수 변화에 따른 평균 메모리 액세스 시간

〈표1〉 고정 우선 순위와 순환 우선 순위 중재의 비교

Rcquest Rate	Worst Memory Access Time		Rcquest Rate	Worst Memory Access Time	
	Rptating Priority	Fixed Priority		Rptating Priority	Fixed Priority
0.002	109	109	0.028	607	1265
0.004	111	112	0.030	639	1554
0.006	120	125	0.032	718	1788
0.008	131	141	0.034	744	1845
0.010	145	161	0.036	750	1882
0.012	206	238	0.038	715	2145
0.014	331	412	0.040	742	2271
0.016	404	667	0.042	715	2498
0.018	476	716	0.044	738	2987
0.020	537	742	0.046	799	4321
0.022	602	861	0.048	781	4795
0.024	550	987	0.050	763	5122
0.026	642	1098			

〈표2〉와 〈그림8〉는 프로세서의 수를 64개, 버스의 수를 8개로 고정하고 메모리 모듈의 수를 변화시켜 얻은 평균 메모리 액세스 시간이다. 〈표2〉는 요구율이 0.02인 경우이다. 〈표2〉와 〈그림8〉에 나타난 바와 같이 메모리 모듈의 수가 감소해 감에 따라서 중재 경쟁에서 승리한 프로세서가 메모리를 점유하려 할때 점유가 불가능할 확률이 높아지므로 다른 프로세서는 한번의 중재 시간 만큼 더 기다리므로 대기 시간이 길어져 결과적으로는 평균 메모리 액세스 기간이 증가한다.

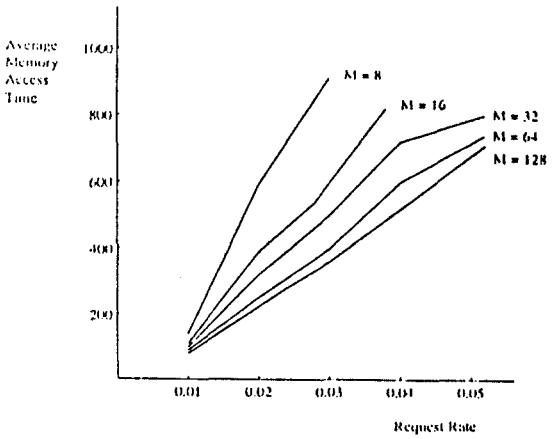
〈표2〉 메모리 모듈수와 평균 메모리 액세스 시간과의 관계

Number of Memory Module	Average Memory Access Time
8	491
16	398
32	336
64	311
128	298

〈표3〉은 프로세서의 수를 64개, 메모리 모듈의 수를 64개로 고정하고 버스의 수를 변경하였을 때 얻어진 버스 이용율이다. 본 시뮬레이션에서는 버스 이용률이란 시뮬레이션 시간 전체에 대한 각 버스가 사용된 시간의 비율로 정의하였다. 버스의 이용률과 선능의 관계는 프로세서의 수, 또는 요구율이 증가함에 따라서 버스에 걸리는 트래픽이 증가하므로 메모리를 요구한

〈표3〉 버스의 수와 버스 이용률의 관계

Number of Buses	Request Rate	Bus Utilization							
		B1	B2	B3	B4	B5	B6	B7	B8
1	0.01	0.9712							
	0.02	0.9745							
	0.03	0.9766							
	0.04	0.9811							
	0.05	0.9875							
2	0.01	0.9245	0.9133						
	0.02	0.9465	0.9377						
	0.03	0.9564	0.9488						
	0.04	0.9642	0.9575						
	0.05	0.9759	0.9641						
3	0.01	0.9165	0.9112	0.9083	0.9054				
	0.02	0.9303	0.9277	0.9265	0.9255				
	0.03	0.9314	0.9300	0.9265	0.9266				
	0.04	0.9317	0.9311	0.9284	0.9273				
	0.05	0.9475	0.9401	0.9344	0.9301				
4	0.01	0.9042	0.9017	0.8812	0.8612	0.8393	0.8002	0.7633	0.7045
	0.02	0.9298	0.9017	0.9263	0.9235	0.9121	0.9100	0.8945	0.8847
	0.03	0.9303	0.9282	0.9278	0.9250	0.9203	0.9199	0.9115	0.8901
	0.04	0.9306	0.9273	0.9250	0.9227	0.09219	0.9198	0.9156	0.8994
	0.05	0.9401	0.9299	0.9295	0.9254	0.9231	0.9209	0.9126	0.9086



〈그림8〉 메모리 모듈수가 성능에 미치는 영향

프로세서의 대기시간은 길어진다. 하지만 버스 이용률이 너무 낮다면 시스템은 필요 이상의 버스를 갖는 것으로도 해석 할 수 있다. 시스템 설계시에 버스 이용률과 대기 시간 또는 메모리 액세스 시간을 고려하여 적정한 버스의 수를 선정하는 고려는 반드시 이루어져야 한다.

시뮬레이션 결과에서는 버스의 번호가 작은 것의 버스 이용률이 높게 나타나는데, 이 이유는 버스번호가 작은 버스부터 차례로 버스를 배정하였기 때문이다. 버스의 수가 적은 경유 한 버스에 걸리는 트래픽이 많아지므로 버스의 이용률이 높은 것으로 나와있다.

위의 성능 분석 결과는 본 논문에서 제시한 중재기와 프로로토콜을 이용한 다중 버스 시뮬레이션을 통하여 얻은 것이다. 기존의 두 단계 중재기를 거치는 방식의 다중버스는 버스를 배정할 때 반드시 메모리의 충돌이 해결된 뒤에 버스를 배정하기 때문에 버스에서의 불필요한 동작이 없다. 하지만 두 단계의 중재기법을 사용하므로 본 논문의 중재 방법에 비해 중재 지연 시간이 길어진다. 따라서 본 논문의 중재 방법에서 메모리 모듈의 수를 적절하게 조정하게 되면 기존의 중재 방법을 이용하는 다중 버스와 비슷한 결과를 얻을 수 있을 것으로 기대된다. 본 논문의 중재 기법은 앞에서도 지적하였 듯이 중재에 사용되는 제어라인의 수를 대폭 감소시켰다는 장점과 중재기의 복잡도를 두단계의 중재에서 한 단계로 줄이므로써 중재

시간을 감소시켰다는 장점을 가지고 있다. 또한 기존의 중재 기법에서는 B-of-M중재기는 구조상 분산이 어려운 반면, 본 논문에서 제안한 중재기는 완전하게 각 프로세서 모듈에 분산된 형태이므로 고장에 강한 특성을 가지고 있다.

Ⅳ. 결론

본 논문을 통하여 제시한 중재기 및 중재 기법은 기존의 중재기 및 중재 기법과 비교하였을 경우 나타나는 장점은 다음과 같이 요약될 수 있다. 첫째, 한 종류의 중재기 만으로 중재가 가능하고 중재 단계를 두 단계에서 한 단계로 줄이므로써 중재기의 복잡도와 중재 시간을 줄였다는 점이다. 둘째, 중재에 이용되는 중재 제어라인의 수를 대폭 감소시킴으로써 비용 및 라인 복잡도를 대폭 감소시킨 점을 들 수 있다. 그리고 마지막으로 완전 분산된 중재기는 고장에 강한 특성을 갖고 있다.

위 특성을 갖는 다중 버스를 시뮬레이션을 통한 성능 분석에서 고정된 프로세서의 수에서 적절한 버스와 메모리 모듈의 수를 결정하여 어플리케이션 특성에 적합한 최적 비용의 다중 버스를 구성할 수 있는 방법을 제시하였다.

앞으로 연구해야 할 과제로는 본 논문에서 제시한 프로토콜의 단점인 불필요한 동작을 감소시키기 위한 방법으로 메모리 모듈의 수를 적절하게 결정하여 기존의 다중 버스와 비슷한 성능을 갖도록 하는 연구가 필요하다.

다중 버스의 구현상의 큰 장애 요인인 버스 라인과 중재기의 복잡도를 본 논문에서 제시한 중재기와 중재 방법을 통하여 해결하였다고 본다. 따라서 본 논문은 병렬 시스템 구조의 다중 버스 설계시 중재기 및 중재 방법에 활용될 수 있을것으로 기대된다.

참고문헌

1) Lamix N.Bhuyan, Qing Yang, and DharmaP.Agrawal, "Performance of Multiprocessor Interconnection Networks", Computer, pp.25-37, Feb., 1989

2) T.N.Mudge,J.P.Hayes, and D.C.Winsor, "Multiple Bus Architecture", Computer, pp.42-48,Jun.,1987.

3) Chita R. Das and Laxmi N. Bhuyan, "Bandwidth Availability of Multiple-Bus Multiprocessors", IEEE, Trans. Computers, Vol.c-34,No.10, pp.918-926, Oct., 1985.

4) Michel Dubois, "Throughput Analysis of Cache-Based Multiprocessors with Multiple Bus", IEEE, Trans. Computers, Vol.37, No.1, pp.58-70, Jan., 1988.

5) Tomas Lang, Mateo Valero, and Ignacio Alegre, "Bandwidth of Crossbar and Multiple-Bus Connections for Multiprocessors", IEEE,Trans. Computers,Vol. c-31, No.12, pp.1227-1234, Dec., 1982.

6) Qing Yang and Safwat G.Zaky,"Communication Performance in Multiple-Bus System", IEEE, Trans. Computers, Vol. 37, No. 7, pp.848-853, Jul., 1988.

7) T.N.Mudge, J.P.Hayes, G.D.Buzzard, and D.C.Winsor,"Analysis of Multiple-Bus Interconnection Networks", Journal of Parallel and Distributed Computing 3, pp. 328-345, 1986.

8) Qing Yang and Laxmi N.Bhuyan, "Performance of Multiple-Bus Interconnections for Multiprocessors", Journal of Parallel and Distributed Computing 8, pp.267-273, 1990.

9) Fayez El Fuibaly, "Design and Analysis of Arbitration Protocols", IEEE, Trans. Computers, Vol. 38, No.2, pp.161-171, Feb., 1989.

10) Cheng-Hsien Tung and C. W. McCarron, "A High Performance Multiprocessor with Partially Orthogonal Multibus and Memory", Proceedings of the ISMM Internaltional Conference, PARALLEL AND DISTRIBUTED COMPUTING, AND SYSTEM, pp.62-66, Oct., 1990.

11) Qing Yang and Laxmi N.Bhuyan, "Design and Analysis of a Decentralized Multiple-

Bus Multiprocessor", International Conference on Parallel Processing, pp.889-892, 1987.

12) R.C.Pearce, J.A.Field, and W.D.Little, "Asynchronous Arbiter Module",IEEE, Trans. Computers, pp.931-932, Sep.,1975.

13) T. Lang and M. Valero,"M-users B-servers Arbiter for Multiple-Buses Multiprocessors", Multiprocessing and Microprogramming, pp.11-18. 1982.

14) D.M.Taub, "Arbitration and Control Acquisition in the Proposed IEEE 896 Futurebus", IEEE, Micro,pp.28-41, Aug., 1984.

15) IEEE Proposed Draft Standard p.896.1, Feb., 14,1990.

16) Laxmi N. Bhuyan, "Analysis of Interconnection Networks with Different Arbiter Desings", Journal of Parallel and Distributed Computing 4, pp.384-403, 1987.

최창훈(Chang Hoon Choi)정회원

1988년 2월 : 명지대학교 전자계산학과 졸업
1990년 2월 : 서강대학교 대학원 전자계산학과(공학석사) 졸업
1990년 1월~9월 : 대우통신 기술개발부 근무
1992년~ 현재 : 서강대학교 대학원 전자계산학과 박사과정 재학중
※주관심분야 : Computer Architecture, parallel processing system

韓昌熙(Chang Hee Han)

1989년 2월 : 서강대학교 전자계산학과 졸업(학사)
1991년 8월 : 서강대학교 전자계산학과 대학원 졸업(석사)
1995년 2월~현재 : 삼성데이타 시스템 품질관리팀

金聖天(Sung Chun Kim)정회원

1975년 : 서울대학교 공과대학 공업교육학(전기전공)학사
1976년~1977년 : 동아컴퓨터(주) Sys.Eng.
1977년~1978년 : 스페리유니맥 Sales Rep.
1979년 : Wayne State Univ.컴퓨터공학 석사
1982년 : Wayne State Univ.컴퓨터공학 박사
1982년~1984년 : 캘리포니아주립대 조교수
1984년~1985년 : 금성반도체(주) 책임연구원
1986년~1989년 : 서강대학교 공과대학 전자계산소 부소장
1989년~1991년 : 서강대학교 공과대학 전자계산학과 학과장
1985년~현재 : 서강대학교 공과대학 전자계산학과 조교수(1985.8~1987.8), 부교수(1987.9~1992.8) 교수(1992.9~현재)
1989년~현재 : 한국정보과학회 병렬처리시스템 연구회 부위원장, 대한전자공학회 및 한국통신학회 논문지 편집위원(1991, 1993), 한국정보과학회 학회지 부위원장(1993)
※주관심분야 : 병렬처리시스템(Parallel Computer Architecture,Interconnection Network), Neural Network, Computer Network