다중 프로세서 시스템에서의 버퍼 및 공유 메모리 최적화 연구

김 종 수[†]·문 종 욱[†]·임 강 빈[††]·정 기 현[†††]·최 경 희[††††]

요 약

고속 입출력 장치를 갖는 다중 프로세서 시스템은 데이터의 처리 성능 향상과 함께 입출력의 집중화에 따른 병목 현상을 줄여줄 수 있다. 이 때 프로세서간의 데이터 전송에 사용되는 공유 메모리는 그 구성과 이용 방법에 따라 시스템 성능에 많은 영향을 미치게 되는데, 본 논문에서는 공유 메모리의 사용방법을 비동기, 메일박스를 통한 인터럽트 전달인지 방식으로 설정한 후 버퍼 및 공유 메모리의 최적 사용량을 예측할 수 있는 모델에 대해 연구하였다. 시스템에 주어지는 입출력 데이터는 이더넷(IEEE 802.3) 망에 흐르는 패킷을 모델로 하며, 이의 대역폭과 burstiness(패킷의 집중화 정도)에 따른 메모리 사용 상황에 대해 살펴보았다. 고속 이더넷(Fast Ethernet) 환경 하에서 시뮬레이션 및 실험에 의해 시스템의 입출력 대역폭뿐만 아니라 패킷의 집중화 정도에 따라서도 버퍼 및 공유 메모리의 사용량이 달라지며, 두 메모리 사이의 사용량에 대한 상관관계가 성립될 수 있음을 알 수 있다.

A Study on Buffer and Shared Memory Optimization for Multi-Processor System

Jongsu Kim[†]·Jongwook Moon[†]·Kangbin Yim[††]
Gihyun Jung[†††]·Kyunghee Choi[††††]

ABSTRACT

Multi-processor system with fast I/O devices improves processing performance and reduces the bottleneck by I/O concentration. In the system, the performance influenced by shared memory used for exchanging data between processors varies with configuration and utilization. This paper suggests a prediction model for buffer and shared memory optimization under interrupt recognition method using mailbox. Ethernet (IEEE 802.3) packets are used as the input of system and the amount of utilized memory is measured for different network bandwidth and burstiness. Some empirical studies show that the amount of buffer and shared memory varies with packet concentration rate as well as I/O bandwidth. And the studies also show the correlation between two memories.

키워드 : 다중 프로세서 시스템(Multi-Processor System), 버퍼(Buffer), 공유 메모리(Shared Memory), 네트웍 대역폭(Bandwidth, Burstiness)

1. 서 론

마이크로 프로세서 및 주변 장치들의 지속적인 성능향상에도 불구하고, 정보 처리율을 더욱 높이기 위해 다중 프로세서 시스템이나 병렬 처리 시스템을 이용하는 연구가 많이 진행되고 있다. 다중 프로세서 시스템은 하나의 프로세서가 모든 일을 처리하던 방식에서 벗어나 여러 프로세서가 작업을 나눠 처리할 수 있도록 하여, 부하 폭주를 방지하고 동시 작업을 수행할 수 있도록 지원한다. 특히 특수 기능에서 고 효율성이 필요한 시스템에서는 특정 시스템 자원의 사용 빈도가 지극히 높게 되는 경향이 있는데, 단일 프로세서의 계산 능력 향상만으로는 해결할 수 없는 문제이므로 다중 프로세서 시스템은 이의 집중화에 따른 병목 현상을 방지할 대책이 되기도 한다[1].

다중 프로세서 시스템의 한 응용은 네트웍 장비에서 찾아 볼 수 있다. 근래 인터넷의 폭발적 사용으로 인해 네트웍에 연결할 수 있는 이더넷 제어 장치가 많이 사용되고 있는데, 이 고속 I/O 장치를 통해 들어오는 데이터들은 빠른 속도와 많은 양 때문에 이들을 처리해 줄 고성능 시스템이 필요해 지는 경우가 많이 발생하게 된다. 이 때 다중

※ 이 논문은 과기부 국가지정연구실사업의 지원으로 연구되었음.
† 준 회 원 : 아주대학교 대학원 전자공학과
†† 정 회 원 : 아주대학교 정보통신전문대학원 전임연구원
††† 정 회 원 : 아주대학교 전자공학부 교수
†††† 정 회 원 : 아주대학교 정보 및 컴퓨터 공학부 교수
논문접수 : 2001년 12월 22일, 심사완료 : 20021년 4월 30일

프로세서 시스템을 통해 입력 데이터를 분산 처리하면 계산 능력과 I/O 사용 능력을 향상시킬 수 있게 된다.

이런 다중 프로세서 시스템에서는 여러 프로세서간의 유기적인 결합을 위한 정보 공유가 필수가 되며, 이 정보 공유를 위한 통신 자원의 성능이 시스템에 큰 영향을 미치는 변수가 된다[3]. 또 이를 구성하기 위한 프로세서간 통신 방식도 시스템 설계 비용에 큰 비중을 차지하는 요인이 되기도 한다.

이러한 시스템에서 자원 공유를 위한 매개체로는 각 프로세서의 시스템 버스 사이를 연결하는 메모리나 I/O 장치들이 이용되는데, 본 연구에서는 버스 사이의 메모리를 연결고리로 사용하는 공유 메모리의 최적화에 대해 논의하며, 특히 많은 데이터 양과 속도, 방향성들을 감안하여 Dual Port RAM의 사용에 초점을 두었다. 이 공유 메모리는 정보 전달 속도에 있어서는 고성능을 발휘하나 용량 당 단가가 비싸다는 단점이 있어 불필요한 낭비를 줄일 수 있다면 성능에는 영향을 미치지 않으면서도 경제적으로 큰 효과를 볼 수 있기 때문에 최적화 노력이 필요한 것이다.

그 연구의 방법으로 다중 프로세서 시스템을 설계할 때 고려하는 각종 매개 변수들(버스 대역폭, 계산능력, 장치접근속도 등)과 입력데이터의 모델(입력 데이터 대역폭, Burstiness 등)을 설정하고 이를 이용하여 이론적인 실험 모델을 설계하였다. 이 연구 모델은 일반성을 지닐 수 있는 구조로 설정되었으며, 이런 매개변수를 조정하여 모의실험을 할 수 있도록 윈도우 기반의 시뮬레이션 프로그램을 제작하였다. 이 프로그램은 각종 변수를 이용하여 사전 제약 사항들을 검토하고 이론적인 결과치를 얻을 수 있도록 작성되었으며, 이 데이터와 실제 실험을 통한 데이터와의 차이를 규명하여 이론적 예측의 타당성을 입증하였다.

서론에 이은 본 논문의 구성은 다음과 같다. 본론의 2장에서는 다중 프로세서 시스템을 구성하는 각종 요소들에 대한 모델과 제한 조건을 정의하고, 3장에서는 이 모델들을 근거로 한 시뮬레이션 프로그램의 구조와 각종 결과 데이터에 대해 분석한다. 4장에서는 실제 다중 프로세서 시스템을 이용하여 직접 실험하고, 이의 결과치와 시뮬레이션에서의 결과치를 비교하여 모델 설정의 정당성을 확인하였다. 마지막 5장에서 이들을 정리하여 전체적인 결론과 향후 연구 방향을 제시한다.

2. 다중 프로세서 모델

이 장에서는 다중 프로세서 시스템의 구조를 프로세서의 연결 모형과 공유 메모리의 사용 방식, I/O 데이터의 접근 형태 등을 통하여 규명하고, 성능 평가는 데이터의 입력에 대한 시스템 내에서의 시간적 접근 방식을 통해 분석하였다.

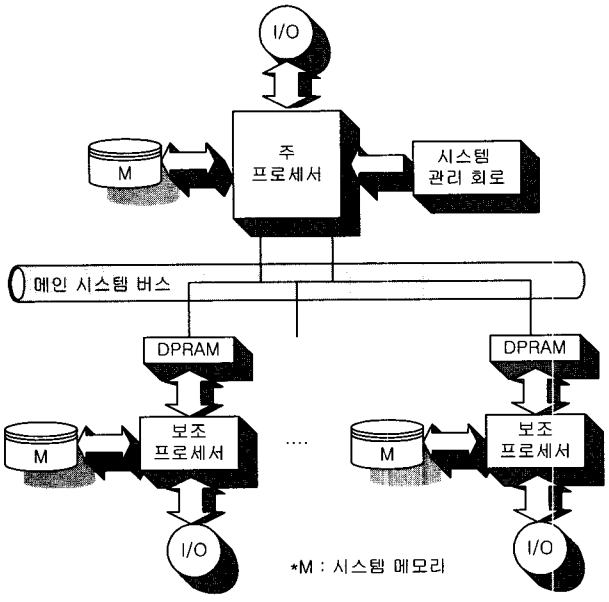
2.1 시스템 구성 모델

다중 프로세서 시스템은 시스템 자원(메모리, I/O 등)의 공유 정도에 따라 Loosely-Coupled 구조와 Tightly-Coupled 구조로 구분되는데 Loosely-Coupled 다중 프로세서 시스템에서는 각각 자체의 자원을 가진 프로세서 시스템들을 특정 통신 매체로 엮고 이들간의 통신을 통해 데이터를 공유하며, Tightly-Coupled 다중 프로세서 시스템에서는 시스템 자원을 각 프로세서가 공유하게 되므로 이를 통해 데이터를 공유하게 된다.

메모리의 입장에서 보면 프로세서는 명령어를 실행하고 데이터를 저장할 용도로 메모리 자원을 필요로 하는데, 이의 구성 위치가 시스템의 특성을 결정짓는 변수로 작용한다[3]. Tightly-Coupled 구조의 시스템에서는 시스템 메모리 자체가 프로세서간의 데이터 공유에 사용되며, Loosely-Coupled 구조의 시스템에서는 프로세서들이 각각 독립적으로 움직일 수 있도록 시스템 메모리가 분리되어 구성되며, 공유 메모리는 프로세서들 간의 정보 전달을 위해 사용되는 일종의 통신 통로로 존재한다[1].

속도와 복잡도의 측면에서 양쪽 다 장단점이 있고, 특히 다중 프로세서 시스템은 적용 대상에 따라 성능의 차가 달라질 수 있기 때문에 본 연구에서와 같이 메모리 공유의 목적이 데이터 전달(현 시스템에서 통신을 위해 사용되는 공유 메모리는 한쪽에서 쓰고 다른 쪽에서 읽어 가는 형태임)일 경우에는 후자의 모델을 선정하는 것이 더 바람직하다.

본 연구에서는 (그림 1)과 같이 자체 자원(메모리, I/O 등)을 갖는 프로세서 시스템들을 일 대 일 방식의 공유 메모리(DualPort RAM)로 엮고, 시스템 버스를 통해 메모리 접근이 가능하도록 하는 Loosely-Coupled 다중 프로세서 시스템으로 구성하였다.



(그림 1) Loosely-Coupled 다중 프로세서 시스템의 구성

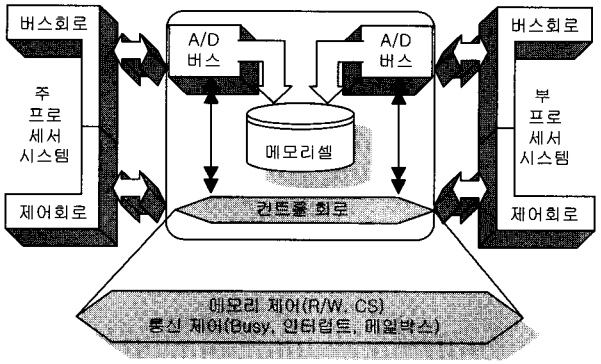
이의 운영을 간략히 살펴보면 다음과 같다. 주 프로세서

는 고속 I/O 데이터를 입력받아 이를 전처리 한 후, 해당 부 프로세서로 전달하기 위해 Dual-Port 메모리에 데이터를 써넣고 데이터 갱신을 알린다. 부 프로세서는 데이터를 공유 메모리를 통해 넘겨받고 이의 정보를 다시 상대측에 전달한다. 이 넘겨 받은 데이터는 후 처리되어 I/O를 통해 다시 출력되게 된다.

2.2 프로세서간 통신 모델

다중 프로세서 시스템의 통신 방식 설계 시 통신 자원의 최소화나 통신 지연을 줄이기 위해 행하는 고려사항 중 전송 모델(병렬, 순차), 전송 타입(동기, 비동기), 전송 프로토콜(FIFO, Multi-Port Memory, Bus block, Bus non-block) 등의 인자는 시스템의 성격을 규정짓는 주요한 요소들이다[3]. 특히 전송 프로토콜 중 버스 방식의 시스템은 다른 연결 방식에 비해 구성을 위한 비용이 적고 따로 필요한 장치들이 없으므로 구현이 용이하며, 시스템 구성에 있어서 융통성이 강한 장점이 있어 상용화되는 많은 다중 프로세서 시스템에 적용되고 있다. 그러나 같은 버스를 통해서만 모든 메모리 자원과 I/O 자원에 접근할 수 있게 되므로 시스템의 전반적인 성능이 버스의 전송 속도와 대역폭에 의해 좌우되는 단점이 있다[10]. 만약 시스템 규모가 커져 일정 버스 대역폭에 도달하게 되면 병목 현상에 의해 시스템 성능이 급격하게 감소되므로 버스 사용권 획득을 위한 지연, 데이터의 충돌 등 시스템 성능 저하의 요인이 될 수 있는 것들을 염두 해 둘 필요가 있다.

이러한 문제로 버스 접속방식에서 직접 연결방식[1]과 아답터 연결방식[2]은 좋은 방법이 아니며, 다단계 상호 연결망이나 이의 변형된 형태의 버스를 사용하는 편이 좀 더 나은 성능을 내게 된다[12]. 이는 장치의 수가 늘어 나도



(그림 2) Dual-port RAM을 이용한 통신설계

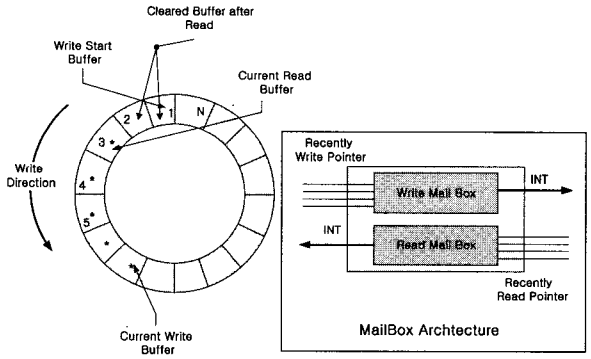
1) 입출력 장치를 버스에 직접 연결하는 방식으로 버스의 순수 전송대역 이외 중재를 위한 통신 부하와 버스 구조에 맞는 인터페이스이어야 한다는 것과 같은 제약사항이 있어 성능이상대적으로 좋지 않음

2) 표준 인터페이스를 통해 시스템 버스에 연결되므로 다중 프로세서 상에서 자원 이용효율의 향상과 인터페이스 수 증가의 이점이 있지만, 일괄적 제어구조를 가지므로 한 장치의 특성이 다른 쪽에 영향을 주게 되는 등의 단점이 있음

대역폭 또한 증가하게 되어 효율저하가 발생되지 않기 때문인데 본 연구에서 사용하는 프로세서의 구조가 I/O 장치의 내부 버퍼링과 지능적 데이터 전송구조를 이용하여 버스 사용을 조절하고 있으므로 버스 대역폭에 영향이 적은 다단계 구조의 형태라고 할 수 있다. 또한 공유 메모리를 사용하여 데이터를 전달하므로 통신으로 인한 시스템 성능 저하 및 버스 충돌 등을 방지할 수도 있다.

본 연구에서는 (그림 2)에 나타낸 것과 같이 정보 전달 속도에 초점을 맞춰 데이터의 병렬 전송을 선택하였으며, 이기종 프로세서간의 연산 및 입출력 속도 차이를 보상하기 위해 전송방식을 비동기로 하였다. 데이터의 흐름은 양방향이 될 수 있도록 하며, 데이터 전송의 신뢰성을 위해 메일박스(mailbox)도 사용하였다. 이 과정에서는 데이터 겹쳐 쓰기와 같은 오류를 방지하고 상대측에 데이터가 전달되었음을 인지시키기 위해 데이터를 전송하는 측에서 데이터를 써넣고, 이의 포인터를 메일박스에 써 넣으면 인터럽트[3]가 발생되어 데이터를 수신하는 측이 공유 메모리 상에 변화가 있음을 알 수 있도록 하였다[7].

Dual-Port 메모리를 사용하는 전략은 통신 포트를 통해 들어오는 데이터를 위해 메모리 버퍼를 N개 구성한 후 이의 포인터와 사이즈 정보를 이용하여 관리하는 방법으로 쓰기 버퍼를 순환(circular) 형태로 사용함으로써 구현할 수 있다[8]. (그림 3)은 N개의 버퍼를 갖는 공유 메모리 상에서 한쪽의 프로세서가 데이터를 써넣으면 다른 쪽의 프로세서가 이를 읽어 가면서 버퍼를 지우고, 서로의 포인터를 넘겨주어 동기화 작업을 수행할 수 있게 하는 링 버퍼의 구조를 나타낸 것이다.

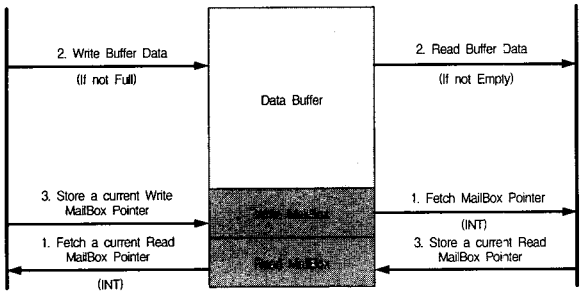


(그림 3) 공유 메모리 내의 링 버퍼 및 메일박스 구조

(그림 3)에서와 같이 Read 포인터는 Write 포인터를 앞지를 수 없으며 Write 포인터 또한 1 회전 후 Read 포인터를 앞질러서 데이터 겹쳐 쓰기 오류가 발생되지 않도록 한다. 이는 Read 포인터와 Write 포인터가 인터럽트에 의해

3) 현재 메시지 전달방식에서는 수신 스케쥴링 정책으로 많이 사용하는 폴링, 인터럽트 I/O 방식 중 상대적으로 구현이 어려우나 성능면에서 뛰어난 인터럽트 I/O 방식을 사용한다[4].

상대측에 자동으로 갱신되도록 설계되어 있어 가능한 것이다. 만약 이 포인터 정보들에 의해 공유 메모리가 다 찬 것으로 판단될 경우 버퍼 메모리에서 공유 메모리로 데이터를 옮기지 못하므로 손실이 발생할 수밖에 없다. 손실 없는 시스템을 구성하려면 입력 데이터에 대한 버퍼링 전략과 예측을 통한 안전한 버퍼량을 알아야 한다.

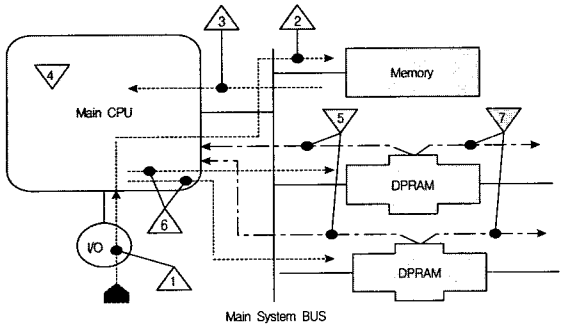


(그림 4) 공유 메모리 접근 순차도

2.3 시스템 타이밍 분석 모델

주 프로세서 시스템과 부 프로세서 시스템은 각각 독립적으로 수행되지만 이들 간의 중계 역할을 하는 공유 메모리의 사용 상황에 따라 각 프로세서들이 영향을 받게 되고, 이는 결국 시스템 전체에 영향을 미치게 된다. 그러므로 데이터가 전달되고 처리되는 과정이 자세히 분석되어야 시스템의 성능과 입력 데이터의 특징에 따른 메모리 사용 모델을 정할 수 있게 되고, 다중 프로세서 시스템의 성능을 향상시키면서도 경제적으로 만들 수 있게 된다. 이를 위해 데이터 입력에서 처리까지의 흐름을 각 프로세서별로 분석하였다.

접근 소요 시간 분석을 위한 주 프로세서 시스템의 데이터 흐름은 (그림 5)와 같으며, 그림의 번호순으로 접근한다.



(그림 5) 주 프로세서 시스템에서의 접근 순서 및 모델

1. I/O 입력 단에서는 데이터 손실을 방지하기 위해 FIFO를 사용하는데, 이의 크기는 입출력 과부하를 견딜 수 있도록 I/O 프로세서가 디자인될 때 고려되며 데이터 흐름에 영향을 주지 않으므로 이 단에서의 접근 시간은 고려하지 않는다[6].
2. 프로세서가 고용량 고속 I/O로부터 데이터를 받아들여 연산 처리를 즉시 할 수 없거나 처리 시간이 길어지게 되면 입력 단에 있는 소량의 FIFO만으로는 데이터의 지속적인 고속 입력을 견딜 수 없게 된다. 이에 따라 많은 경우 대용량 시스템 메모리(버퍼 메모리) 등으로 데이터를 이동시켜 누락을 방지하는 방법이 가장 주효하다. 이런 단순 이동 용도에는 속도와 기능면에서 DMA전송 방식을 이용하는 것이 효과적이다. 이에 소요되는 시간을 결정하기 위해, 프로세서에 사용되는 시스템 클럭을 S_{CLK}, 명령어 처리의 기준이 되는 Cycle당 소요 시스템 클럭 수를 C_{CLK}, 한 Word당 전송을 위한 Burst Access Cycle 수를 Cb_{cycle}, Burst당 전송 Word를 C_{word}, Bus Width를 C_{width}(bit)라고 설정하면 DMA를 기동하여 FIFO로부터 시스템(버퍼) 메모리로의 Burst Read(Write)하기 위한 접근 시간, T_{bst}[4]는 $T_{bst}=\frac{1}{S_{CLK}}\times Cb_{cycle}\times C_{CLK}$가 되고, bit당 Burst DMA를 통한 메모리 접근 시간, T_{bstpb}는 $T_{bstpb}=\frac{T_{bst}}{(C_{word}\times C_{width})}$가 된다. 결과적으로 DMA를 통해 I/O 입력 데이터를 시스템 메모리에 저장하는데 소요되는 시간, $T_{writeMEM}$ (T_2)은 다음과 같다.

$$T_{writeMEM}=T_{bstpb}\times n \quad (n: \text{I/O 입력 bit 수}) \qquad (1)$$

3. 주 프로세서는 버퍼 메모리에 넣어진 데이터를 읽어 특정 전 처리 작업을 수행하게 되는데, 이를 위해 대모리로부터의 데이터 패치 작업이 필요하다. 일반적으로 프로세서 내부의 데이터 캐쉬를 이용하면 처리 속도가 증가하지만, 빠른 외부 입출력 자원을 이용하여 처리하는 시스템에서는 정보 변경이 잦게 되어 이의 효과를 기대할 수 없으며, 또한 CPU는 작업 수행을 위해 데이터를 개별적으로 읽어 들이게 된다. 그러므로 이의 시간을 분석하기 위해 Word당 전송을 위한 Single Beat Access Cycle수를 Cs_{cycle}이라고 하면, CPU에 의한 시스템 메모리의 Single Beat Read(Write) 접근 시간(T_{sing})은 $T_{sing}=\frac{1}{S_{CLK}}\times C_{Sycle}\times C_{CLK}$가 되고, bit당 Single Beat 메모리 접근 시간(T_{singpb})과 Single Beat 접근을 통해 데이터를 프로세서가 처리 할 수 있도록 읽는 시간, $T_{readMEM}$ (T_3)는 각각 다음과 같게 된다.

$$T_{singpb}=\frac{T_{sing}}{C_{width}},\quad T_{readMEM}=T_{singpb}\times n \qquad (2)$$

4. 프로세서에서 읽어 들인 데이터를 처리할 때에는 일정 단위의 I/O 데이터에 대하여 수행되는 명령어 사이클과 데이터마다에 대하여 수행되는 명령어 사이클이 필

4) Burst Read, Burst Write, Single Beat Read, Single Beat Write 모두 메모리 접근 속도가 다르나 최악의 경우(Worst Case)등을 고려하여 Read, Write를 평균하여 계산한다.

요하다. 프로세서의 구조에 따라 명령어 처리 형태가 달라 명령어 당 처리 사이클 수가 틀리므로 이를 일반화 할 수 없다. 또한 수행되는 프로그램의 길이 등도 수행시간을 결정하는 변수가 되므로 데이터 처리 시간인 $T_{exe}(T_4)$는 단위 데이터 당 수행하는데 걸리는 시간(T_{exedef})과 bit당 데이터를 처리하기 위한 시간, T_{exepb}로써 나타낸다.

$$T_{exe} = T_{exedef} + T_{exepb} \times n \tag{3}$$

5. 동기화 과정에서 발생하는 시간(T_{sync} (T_5))을 살펴보면 변화된 데이터의 위치를 알려주기 위해 공유 메모리 내의 특정 영역(메일박스)에 정보를 써 넣는 시간, 인터럽트를 인지 후 정보를 읽어 들이는 시간 등이 포함된다. 이 동기화 기능은 프로세서 명령 처리 시간과 같이 일정한 I/O 입력 길이 패턴 (예, 패킷 크기)마다 발생한다. 데이터 처리 시간과 동기화 처리 시간에서 다른점은 데이터 처리시간에서는 I/O 데이터의 크기와 상관 없이 수행되는 시간과 비례적으로 증가되는 시간이 있는 반면 동기화 처리시간에서는 데이터 길이에 상관없이 고정적인 처리 시간을 갖는다는 것이다. 이 처리 과정 역시 프로세서의 인터럽트 명령어 처리 부분이므로 필요 사이클 수를 계산하면 된다.
6. 공유 메모리에 접근하여 정보를 써 넣는 것은 시스템 버스를 통하여 일반 메모리에 접근하는 것과 같은 방식이어서, 메모리의 특성과 접근 시간에 의해 결정된다. 보통 SRAM과 같은 구조이므로 Single Beat Read (Write)를 수행하는데(T_{singDP}) 디바이스의 속성에 따른 접근 과정 사이클 수만 다를 뿐이다. 접근 방식은 같고 접근 사이클 수만이 틀림을 감안하여 계산하면, bit당 Single Beat 메모리 접근 시간, $T_{singDPpb}$과 공유 메모리로의 데이터 전송 시간, $T_{writeDP}(T_6)$는 다음과 같다.

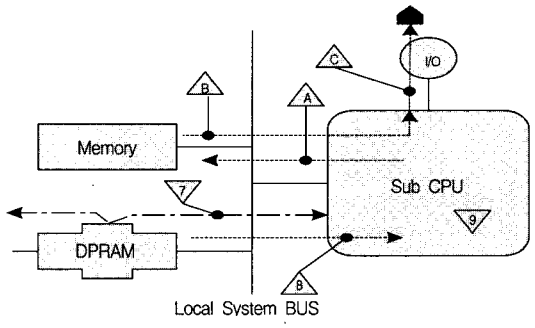
$$T_{singDPpb} = \frac{T_{singdp}}{C_{width}}, \quad T_{writeDP} = T_{singDPpb} \times n \tag{4}$$

그러므로 식 (1)에서 식 (4)에 의해 주 프로세서 시스템에서 I/O로부터 데이터가 공유 메모리에 전달되는데 걸리는 시간(T_{ALL})은

$$\begin{aligned} T_{ALL}(M) &= T_2(M) + T_3(M) + T_4(M) + T_5(M) + T_6(M) \\ &= T_{writeMEM}(M) + T_{readMEM}(M) + T_{exe}(M) \\ &\quad + T_{sync}(M) + T_{wiiteDP}(M) \end{aligned} \tag{5}$$

이 된다(M은 주(Main) 프로세서 시스템을 뜻한다).

부 프로세서 시스템에서도 역시 같은 자원들을 가지고 있으므로 같은 방식의 데이터 접근이 이루어진다. 데이터의 흐름은 주 프로세서 시스템에 반대가 되는, 공유 메모리에서 I/O로의 이동이지만 접근 시간 분석은 같은 방식으로 이루어 진다. 단, 버스 속도, 주변장치에 대한 접근 시간 등이 달라지므로 관련된 변수 값 등은 다른 내용을 갖게 될 것이며, 데이터는 (그림 6)의 7, 8, 9, A, B, C의 순으로 흐르게 된다.



(그림 6) 부 프로세서 시스템에서의 접근 순서 및 모델

부 프로세서 시스템 모델에서 공유 메모리와 시스템 메모리의 접근 방식이 주 프로세서 시스템과 같으므로, 접근 시간을 산출하는 방법 역시 같다. 단, T_{exe}는 부 프로세서 시스템에서의 데이터 처리 시간으로써 프로세서의 명령어 처리 방식에 의해 결정된다. 이에 따라 부 시스템에서 공유 메모리로부터 I/O로 데이터가 전달되는데 걸리는 시간(T_{ALL})은

$$\begin{aligned} T_{ALL}(SK) &= T_7(SK) + T_8(SK) + T_9(SK) + T_A(SK) + T_B(SK) \\ &= T_{sync}(SK) + T_{readDP}(SK) + T_{exe}(SK) \\ &\quad + T_{writeMEM}(SK) + T_{readMEM}(SK) \end{aligned} \tag{6}$$

이다(Sk은 k번째 부(SUB) 프로세서 시스템을 뜻한다)

한편, n bit를 처리하기 위한 시간이 T_{ALL} 이었으므로 초당 데이터 처리량은 각각

$$\begin{aligned} &Throughput(M\_or\_Sk)(bps) \\ &= \frac{1(s)}{T_{ALL}(M\_or\_Sk)(ns)} \times n(bit) \end{aligned} \tag{7}$$

이 되며, 이를 통해 시스템의 처리 가능한 대역폭을 산출할 수 있게 된다. 이는 무리한 입력으로 손실을 발생시키기보다 입력 데이터의 대역폭(Bandwidth)을 제한하는 방법 등으로 시스템을 안전한 영역에서 동작시킬 수 있도록 하는데 중요한 정보가 된다.

2.4 입력 데이터의 모델링

시스템의 성능을 측정할 때 입력 데이터의 특성(형태, 크기 등)은 처리 결과에 많은 영향을 미치게 된다. 이에 본 연구에서 사용하는 다중 프로세서의 입력 모델에 다음과 같은 제한 조건을 갖도록 하였다.

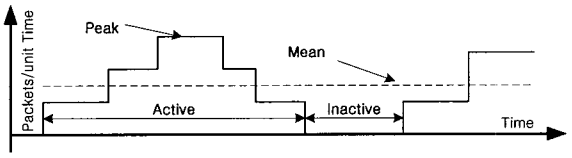
전제 1. *최대 대역폭을 제한할 수 있으며, 일정한 값을 갖는다.*

전제 2. *데이터는 크기는 일정하며, 같은 패턴을 가지고 지속적으로 전달된다.*

전제 3. *입력 단에서는 데이터 충돌 등 외부요인에 의한 영향을 받지 않는다.*

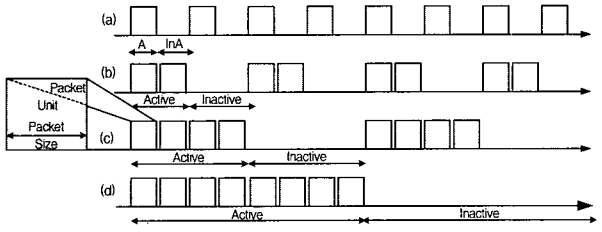
전제 4. *입력 데이터의 형태는 해당 I/O 장치에서 연속된 데이터 묶음으로 구성되는 패킷을 기반으로 한다.*

이와 같은 입력 환경 하에서 데이터 패턴을 변화시키며 이에 대한 영향을 측정한다.



(그림 7) Burstiness 변수의 구성([14]참조)

입력 대역폭이 결정되면, 해당 대역폭 내에서 burstiness 변수를 도입한다. 이에 대한 정의는 여러 부분에서 사용 방식과 표현을 달리하지만, 그 공통적 의미는 "데이터 흐름의 혼잡을 나타내주는 척도"로 사용된다. 그 중 주로 사용되는 표현으로는 PBR(peak bit rate : 짧은 주기 동안의 연속된 bit열 비율), MBR(mean bit rate : 긴 주기 시간 동안의 평균 bit 비율)을 이용하여 이들 사이의 비율로써 나타내는 peak duration 측정 방법과 데이터가 흐르는 주기와 흐르지 않는 주기를 비율로써 나타내는 busy to idle period 측정 방법 등이 있다[11](그림 7). 본 연구에서는 이들을 조합한 Burstiness의 개념을 사용한다.



(그림 8) 패킷에 대한 Burstiness 표현

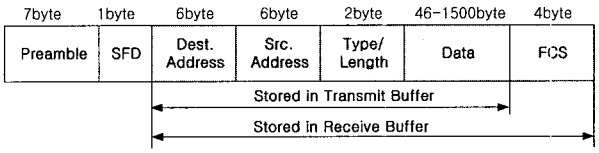
(그림 8)은 시간에 따라 I/O 장치에 흐르는 패킷 기반의 데이터 형태를 나타내고 있다. 짙은 색의 부분이 데이터가 전달되는 부분이며, 패킷의 길이는 I/O 장치로 입출력 되는 한 개의 데이터 패킷의 크기를 나타낸다. 여기에서는 I/O 입출력 장치가 가지는 최대 대역폭의 50%에 해당하는 대역폭을 예로 표시하고 있다. (a)에서는 데이터 전달 기간(Active time)과 휴지 기간(Inactive time)이 반반씩 고르게 분포하여 전달되고 있는데(Uniform Transfer), (d)로 갈수록 패킷의 집중화 정도가 커지고 있다. 그러나 4가지 경우 모두 긴 시간으로 보아서는 같은 대역폭(Max. I/O Bandwidth * 50%)을 가지고 있다. 이를 위에서 정의한 burstiness로 표현하면, 4가지 모두 긴 시간에서 측정하는 MBR은 같지만 짧은 시간에서 측정하는 PBR은 (a)보다 (d)가 크므로 burstiness도 (a)보다 (d)가 크게 된다. 전제에서 같은 형태의 데이터 패턴이 입력된다고 하였으므로 I/O 장치의 대역폭은 데이터 전달시간과 휴지시간의 비로 표시할 수 있으며, burstiness는 같은 대역폭 내에서의 데이터 집중화 정도로 표시한다.

$$B(\text{Burstiness}) = (\text{Bandwidth, Concentration})$$
$$= \left(\frac{(\text{Active})}{(\text{Active} + \text{Inactive}))} \times \text{Max.of I/O B.W.}, \frac{\text{Concentrate } n}{\left(\frac{\text{I/O B.W.}}{\text{Packet Size}} \right)} \right) \quad (8)$$

3. 시뮬레이션

2장에서 설명한 실험모델의 수식 계산과 실제적인 움직임을 파악하기 위해 FIFO, 버퍼, DPRAM의 메모리 사용량을 가상으로 실험하는 프로그램을 작성하였다. 이 프로그램을 이용하여 이론에 의한 실험 예상치 데이터를 산출한다.

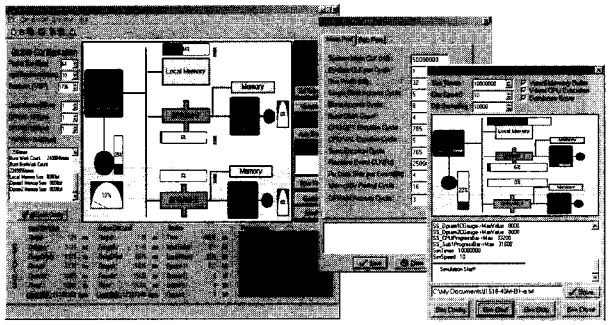
본 연구에서 시스템의 성능을 측정하기 위해 시뮬레이션과 실험에서 사용한 입력 데이터의 형태는 이더넷 기반의 패킷으로 설정하였다. 이더넷은 근래 고속 입출력 방식(Fast Ethernet)으로 많이 사용되고 있으며, 표준화된 형태를 가지면서도 패킷의 크기 조정이 자유로워 실험의 특성을 실험하기에 적합한 구조를 가지고 있다.



(그림 9) 이더넷 패킷 구조

(그림 9)은 이더넷 패킷의 구조를 나타내고 있다. 이더넷 구조상 패킷의 크기는 최소 64Byte에서 최대 1518Byte로 구성될 수 있고(Preamble, SFD 제외), 패킷 간격(Inter Packet Gap)이 존재한다. 망에서 데이터는 패킷 단위로 움직이며, 패킷 내에서의 데이터 흐름은 망의 최고 속도에 동기하여 흐른다. Preamble 및 SFD(Start Frame Delimiter)는 패킷의 시작을 알리는 것으로 망에 흐름은 있으나 접속장치의 입력 데이터로는 저장되지 않는다. 패킷의 크기가 작을수록 이 비 데이터의 비율이 상대적으로 크게 되므로 시스템 성능 측정에 이의 영향이 크게 미치게 된다. 이 비 데이터 영역에 대한 영향을 시뮬레이션에 반영하기 위해 패킷의 입력 모델에 8Byte가 더해지게 되고, 이 시간 동안에는 FIFO로의 입력은 없도록 하였다. 시뮬레이션에서는 입력 모델에 대한 변수로써 패킷의 크기, 입출력 속도, 해당 속도에서의 Burstiness를 변경할 수 있도록 하였다.

(그림 10)은 제작된 시뮬레이션 프로그램으로 직접 볼 수 있도록 설정한 후의 모습이다. 이 프로그램은 Inprise 사의 C++ Builder 툴을 이용하여 C++언어와 Borland Lib.으로 작성되었다. 이 프로그램은 실제 상황의 1/100,000~1/10배 속도로 동작하며 각종 변수들을 변화시키면서 자동으로 데이터를 산출하고 기록을 남길 수 있도록 되어 있는데, 기록에는 시스템 모델 변수에 의해 변화되는 항목과 시간, 각 메모리의 크기 등의 정보가 저장된다.



(그림 10) 가상 시뮬레이션 주 출력 화면

이 시뮬레이션에서 사용하는 각종 시스템 변수는 실제 실험과의 비교를 위해 4장에서 사용할 실제 시스템의 설정 값을 이용한다. 이를 위해 주 시스템으로는 PowerPC 코어를, 부 시스템으로는 ARM7TDMI 코어를 사용한 다중 프로세서 시스템의 각종 변수들을 설정 값으로 이용한다. 이 값은 데이터 쉬트, 프로세서 참고자료 등을 바탕으로 설정한 값이다(그림 11).

주 프로세서 시스템 분석을 위해 각종 변수를 식 (1)에서 식 (5)까지 적용하여 계산하면

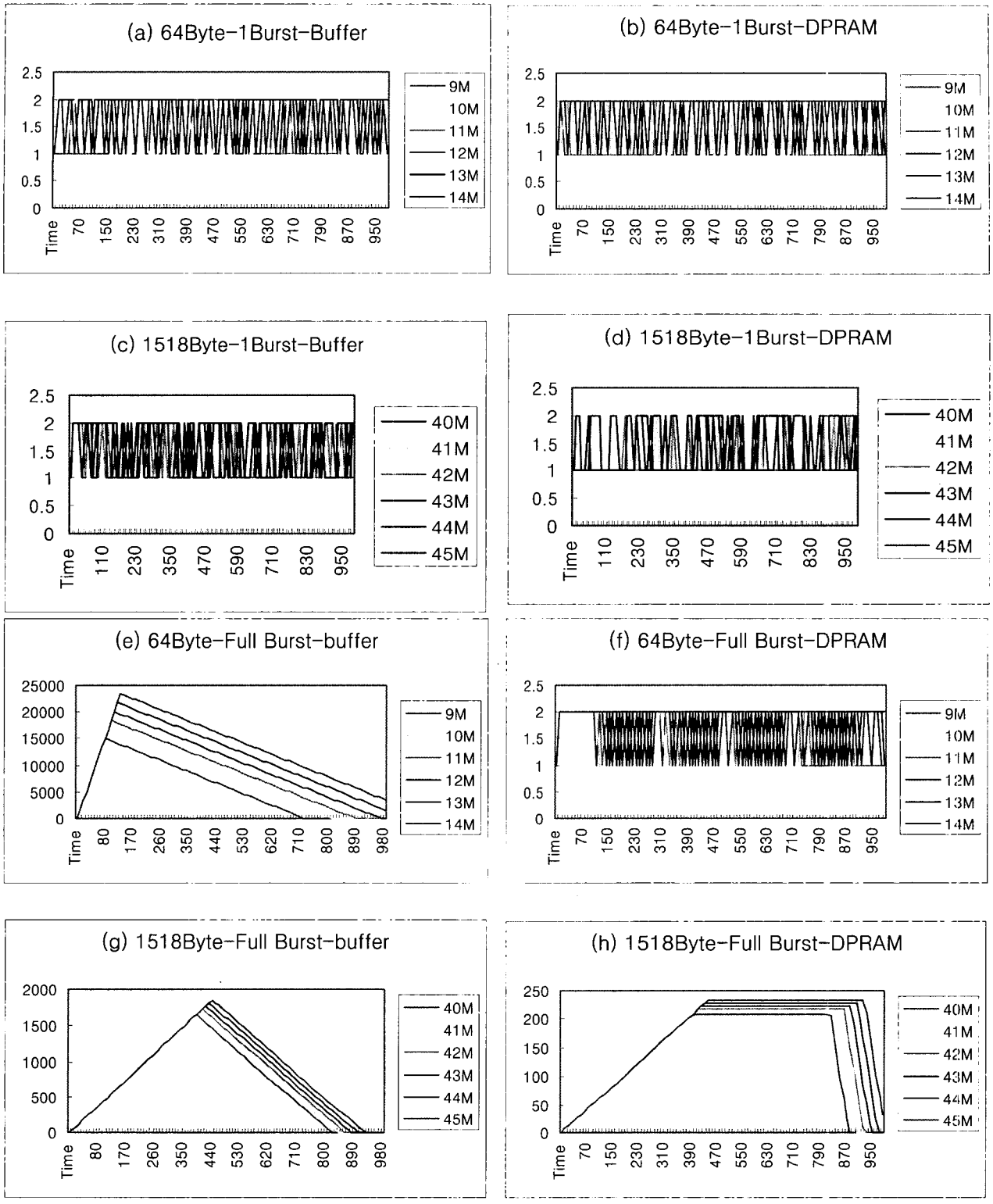
1. 시스템 클럭, S_{CLK} = 50MHz = 50,000,000Hz
2. 파이프 라인 구조의 명령어 처리기와 명령어 캐쉬의 기동에 의해 C_{CLK} = 1
3. Burst Access Cycle 수는 주 메모리로써 SDRAM을 사용하므로 (5-1-1-1) 구조로의 4 Word 전송 전략으로 인해 $Cb_{cycle,}$ = 8, Burst 당 전송 Word, C_{word} = 4
4. SDRAM으로의 Single Beat Access Cycle 수는 1 Word 전송 당 $Cs_{cycle,}$ = 5
5. 공유 메모리(DPRAM)로의 Single Beat Access Cycle 수는 1 Word 전송당 $Cs_{cycle,}$ = 3
6. CPU의 외부 버스 인터페이스, C_{width} = 32bit(최대로 설정하여 최대의 성능을 구현)
7. DMA를 기동하여 FIFO로부터 시스템 메모리로의 Burst Read(Write) 시간, T_{bst} = 160ns 이고, bit당 Burst DMA를 통한 메모리 접근 시간, $\boldsymbol{T_{bstpb}}$ = **160/128 = 1.25ns**
8. 주 메모리에서 CPU가 데이터를 읽어 오는 시간인 Single beat 접근 시간, T_{sing} = 100ns이고, bit당 Burst DMA를 통한 메모리 접근 시간, $\boldsymbol{T_{singpb}}$ = **100/32 = 3.125ns**
9. CPU가 데이터를 공유 메모리에 써 넣는 시간은 Single beat 접근 시간으로, T_{singDP} = 60ns이고, bit당 Burst DMA를 통한 메모리 접근 시간, $\boldsymbol{T_{singDPpb}}$ = **60/32 = 1.875ns**
10. 데이터를 처리하는 루틴에서 사용된 명령어 중 데이터 패킷 당(크기에 상관없이) 고정적으로 사용되는 명령어 수는 305개이고, 패킷 내의 한 Word당 필요한 명령어 수는 22개 이다. 고정 사용 명령어 개수중 45개는 Jump 명령어로서 파이프라인이 깨지게 되는 데 PowerPC 코어에서는 분기 예측 기법을 이용하여 1개의 명령어 분만 손해를 보게 된다. 그러므로 이에 대해 2배의 처리 사이클이 필요하게 되며, 120개는 시스템 메모리에서 데이터를 Read, Store 하는 명령이므로 SDRAM으로의 Single beat Access Cycle이 필요하게 된다. 이에의해 총 수행되는 명령어 사이클은 45 * 2 + 120 * 5 + 140 = 830로 $\boldsymbol{T_{exedef}}$ = **830 * 20ns = 16600ns**이고, bit당 데이터를 처리하기 위한 평균 시간, $\boldsymbol{T_{exepb}}$ = **22/32 * 20ns = 13.75ns**
11. 공유 메모리 동기화를 위해 사용되는 인터럽트 처리 루틴에서는 필요한 명령어 처리 개수가 295개이고 이 중 130개가 Read/Store용 명령, 15개가 Jump 명령이 된다. 그러므로 15 * 2 + 130 * 5 + 150 = 830 사이클이 필요하게 되어 동기화에 소요되는 시간, $\boldsymbol{T_{sync}}$ = **830 * 20ns = 16600ns**
12. 이에 의해 주 시스템에서 I/O로부터 데이터가 공유 메모리에 전달되는데 걸리는 시간 $T_{ALL}(M)$ = (T_{bstpb} + T_{singpb} + $T_{singDPpb}$ + T_{exepb}) * n + T_{exedef} + T_{sync} = **(1.25 + 3.125 + 1.875 + 13.75) * n + 16600 + 16600 = 20 * n + 33200**

같은 방식으로 부 프로세서 시스템의 설정 값(다른 항목은 동일, 단위 데이터당 필요 사이클 750Cycle, Word당 필요 사이클 25Cycle, 동기화 시의 사이클 765Cycle)으로 식 (7)을 이용하여 데이터 처리 성능을 예측하면 <표 1>과 같다.

〈표 1〉 이론적 계산에 의한 데이터 처리량

		입력 패킷 크기	
		64Byte(최소치)	1518Byte(최대치)
시스템	주 시스템 (Throughput(M))	11,786,372bps	43,987,252bps
	부 시스템 (Throughput(S1))	12,337,349bps	41,033,960bps

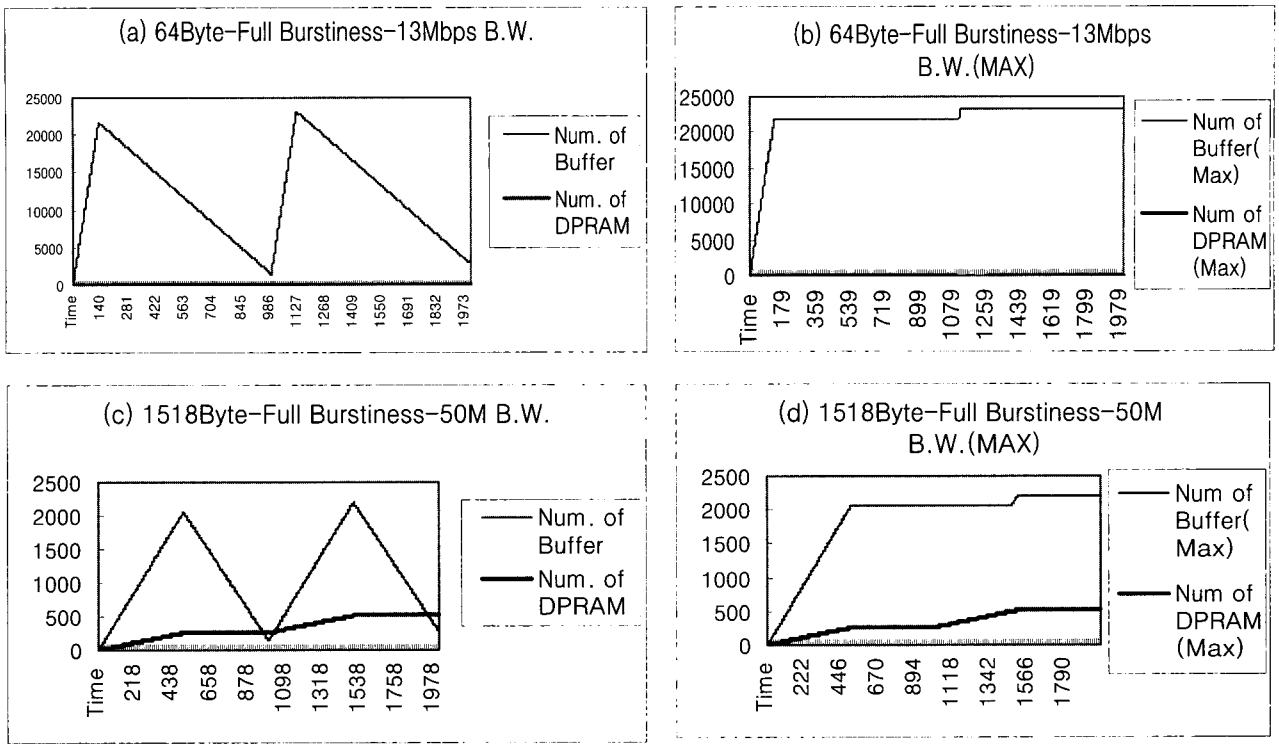
위의 표에 의해 이 시스템에서는 입력 패킷의 크기가 작을



(그림 11) 입력대역폭에 따른 버퍼 및 공유 메모리 사용량

때는 부 프로세서의 성능이 더 뛰어나고, 클 때는 주 프로세서의 데이터 처리 능력이 더 뛰어 나다는 것을 알 수 있다.

입력 데이터의 대역폭을 가변하며 시뮬레이션하면 (그림 11)과 같은 메모리 사용 분포를 볼 수 있는데, 8개의 그림에서 x축은 경과 시간(ms 단위)을, y축은 버퍼나 공유 메모리의 사용량을, 각 목차들은 입력 대역폭을 나타낸다. 또 (a)~(d)는 고른 입력, (e)~(f)는 가장 burstiness가 높은 입력에 대한 반응을 표시한 것이다.



(그림 12) 처리능력(예상치) 이상의 입력에 대한 반응

이 시뮬레이션에서는 입력 데이터 모델 중 최소(64Byte) 및 최대(1518Byte) 크기의 패킷에 대해 시스템의 예상 데이터 처리 능력치 근처의 입력 대역폭을 주고 이에 대한 메모리 자원의 사용량을 산출한 것이다. 이 분포에서 파악되는 것은 크게 3가지이다. 첫째, 입력 대역폭이 시스템의 주 부 프로세서 시스템의 최소 처리 속도보다 크다면 무한 루틴에서는 버퍼의 크기가 계속적으로 증가하게 되므로 데이터 손실이 일어나게 됨을 알 수 있다((그림 12) (b)참조).

Min(Throughput(M), Throughput(Sk) < Input B.W.,
Buffer → ∞

둘째, 데이터의 흐름에 있어 선행 처리를 하는 프로세서 시스템 보다 후 처리를 하는 프로세서 시스템의 처리속도가 높다면 공유 메모리의 사용 개수는 항상 2이하이다(64Byte 패킷 크기에서의 처리성능이 주 프로세서 < 부 프로세서임 (<표 1> 참고)).

$$Thrughput(M) < Throughput(Sk) \rightarrow \text{DP RAM} \leq 2$$

셋째, Burstiness가 없는 경우 입력 대역폭이 시스템 보다 낮게 되면 하나의 패킷이 들어와서 처리되는 시간이 다음 패킷이 들어오기 전까지의 시간 보다 같거나 짧게 되므로 실시간 패킷 처리와 같은 형태가 된다. 이 때는 버퍼와 공유 메모리의 크기가 각각 2 이하가 된다. ((그림 13) (a)~(d) 참고) 역으로 Burstiness를 해당 대역폭 내에서 최대가 되도록 설정하면 버퍼 메모리와 공유 메모리의 사용 크기를 결정할 수 있게 된다.

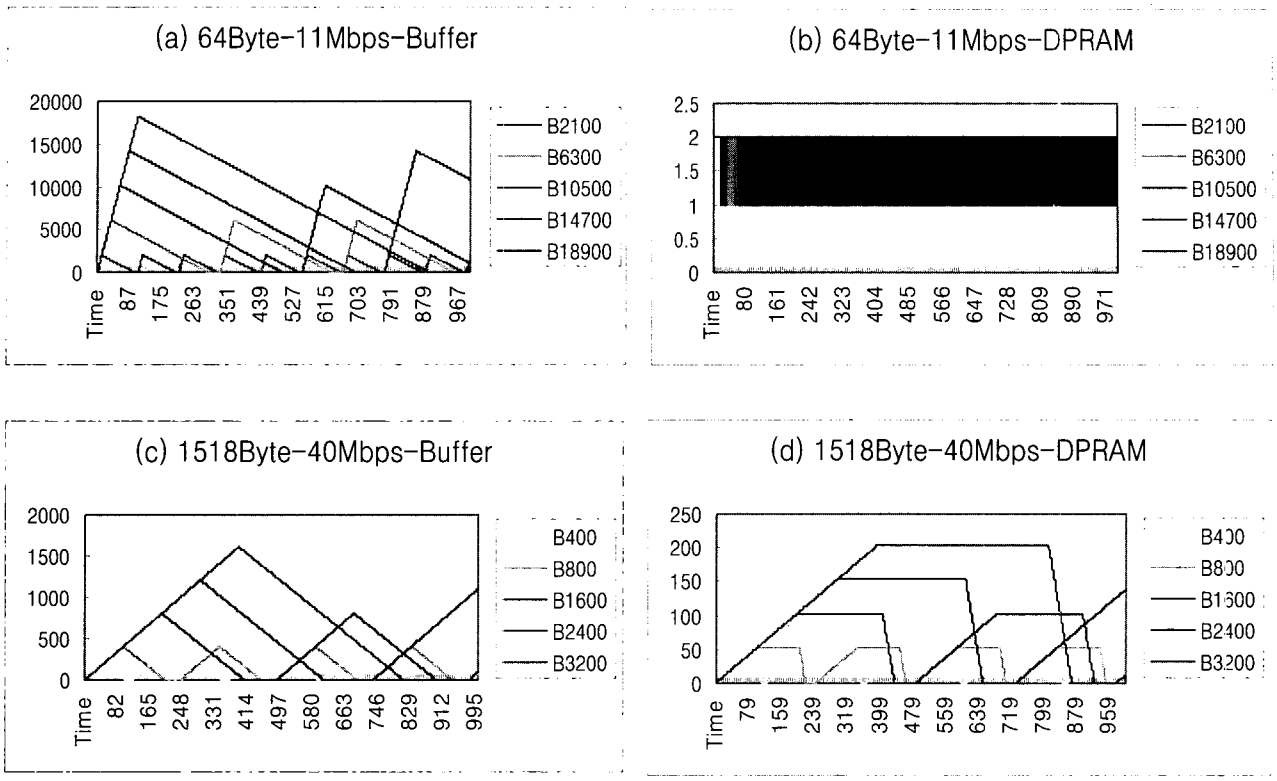
Limited B.W.& Burstness(1) → Buffer ≤ 2, DPRAM 2

(그림 12)는 시간(x 축 ; ms 단위)이 경과함에 따라 버퍼 및 공유 메모리의 사용량(y 축 ; 버퍼 개수) 추이를 나타낸 것으로, 처리 능력 이상의 대역폭을 입력으로 받아들이게 되면 (a), (c)에서와 같이 버퍼가 완전 소진되지 못하고 처리량과 입력량의 차이만큼이 남아 있게 되는 것을 확인할 수 있다. 한편, 공유 메모리의 관점에서 보면 주 프로세서 보다 부 프로세서의 처리능력이 큰 64Byte 입력 패킷의 경우, 필요한 공유 메모리의 양이 2개 이하이고, 반대로 처리 능력이 주 프로세서가 좀 더 나은 1518Byte 입력 모델에서는 주·부 프로세서 시스템의 처리 능력 차만큼 공유 메모리가 증가되고 있음을 확인할 수 있다. (b), (d)는 시스템에서

〈표 2〉 입력 데이터의 변수, Burstiness

	입력 패킷 크기	
	64Byte(최소치)	1518Byte(최대치)
입력할 평균 대역폭	11Mbps	40Mbps
Burstiness 가변치[5]	1/19097~1	1/3276~1

5) burstiness는 0(Uniform)~1(Max. Burst) 사이의 값으로 설정되나 측정의 편의상 그림 등에서는 (1/19097~1)대신 (1~19097), (1/3276~1)대신 (1~3276)의 수치를 사용한다.



(그림 13) 입력대역폭에서의 Burstiness에 따른 반응

사용되었던 최대 버퍼 및 공유 메모리의 수를 나타내는 것으로, 시스템에서 데이터 손실이 발생하지 않도록 하기 위해서는 최대 사용량 만큼의 메모리가 확보되어야 하므로 이를 위한 측정값으로 표시한 것이다. 위 상황의 경우 시간이 지속됨에 따라 계속적으로 버퍼 메모리가 증가하게 되어 버퍼 및 공유 메모리의 사용량이 결정될 수 없게 된다.

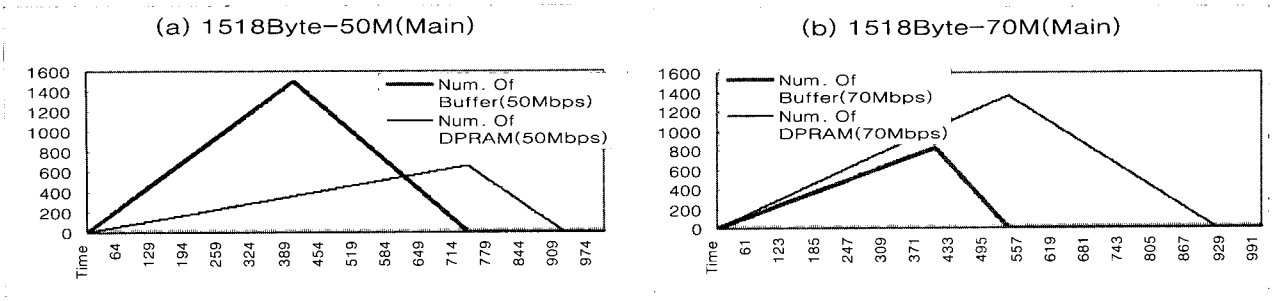
결국, 데이터 손실이 없는 상황에서 공유 메모리의 크기가 시스템에 영향을 받는 경우는 입력 대역폭이 주부 프로세서 시스템 보다 작으면서 부 프로세서 시스템의 처리 성능 합이 주 프로세서보다 낮고, 해당 입력 대역폭에서의 burstiness가 존재하는 때로 정의될 수 있다.

한편, 두 메모리간의 사용량 관계를 규명하기 위해 앞에서 설정한 환경 하에서, 처리능력 이하의 평균 대역폭을 입력으로 하여 패킷의 burstiness에 대한 메모리 사용량을 확인한다. 이 때 64Byte 입력 패킷에 대해서는 11Mbps, 1518Byte에 대해서는 40Mbps의 평균 대역폭을 실험치로 책정하고, 이의 대역폭 내에서 패킷의 burstiness를 가변하며 사용량을 측정한다.

(그림 13)에서는 각각 64, 1518Byte에 대한 버퍼, 공유 메모리 사용량을 나타내고 있다. (x축-시간 경과, y축-버퍼 사용량, 항목-Burstiness) 각 상황에서 burstiness를 가변하며 실험한 결과, 이에 따른 버퍼 메모리의 최대 사용량을 결정할 수 있었다(a, c). (a)에서는 burstiness가 증가될수록 버퍼의 사용량 모양이 큰 삼각형을 이루고 있는데, 삼각형의 꼭대기가 시간 상으로 보면 Burst 한 데이터 전송의 끝임 알 수 있다. (d)의 경우 사다리꼴 모양을 나타내는데, burstiness가 증가될수록 큰 사다리꼴을 나타내며 공유 메모리 사용량이 일정 위치에서 같은 크기로 지속되는 것은 버퍼에서 데이터를 소진하고 있음을 나타내 주고 있다.

주·부 프로세서 시스템의 성능 차이에서 부 프로세서 시스템이 우세한 64Byte 시뮬레이션에서는 공유 메모리 사용량이 역시 2이하 였으며(b), 주 프로세서 시스템이 우세한 1518Byte에서는 메모리 사용량의 최대치를 측정 할 수 있었다. 이 시뮬레이션을 통해 입력 데이터의 burstiness가 버퍼 및 공유 메모리 사용량에 영향을 미치고 있음을 확인할 수 있다.

앞의 시뮬레이션에서는 메모리 사용량을 측정하기 위해 버퍼와 공유 메모리 양쪽에 메모리 자원을 충분히 사용할 수 있도록 설정한 후, 결과값을 측정한 것이다. 이의 경우 주부 프로세서 시스템의 데이터 처리 능력보다 작은 입력 대역폭이면 메모리 사용량이 유한함이 증명되었다. 그러나 이렇게 메모리 크기가 제한되지 않은 상태에서는 주부 프로세서 시스템 각각의 처리 성능에 따라 메모리 사용량이 결정되므로 시스템의 안정성에는 문제가 없지만 경제성에는 문제의 여지가 있게 된다. 공유 메모리 최적화 설계가 목표가 된 이유 역시 버퍼 메모리에 비해 상대적으로 값이



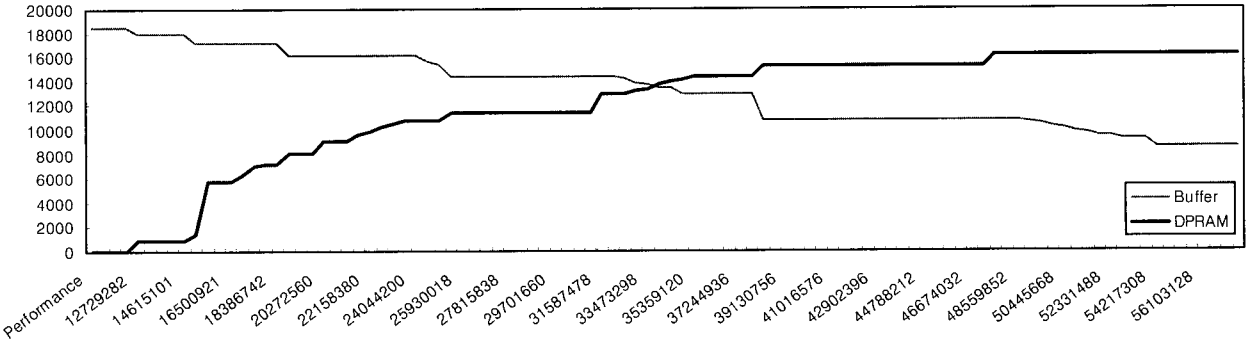
(그림 14) 주 프로세서 성능에 따른 메모리 사용량

비싼 공유 메모리의 사용량을 줄이기 위함이었다.

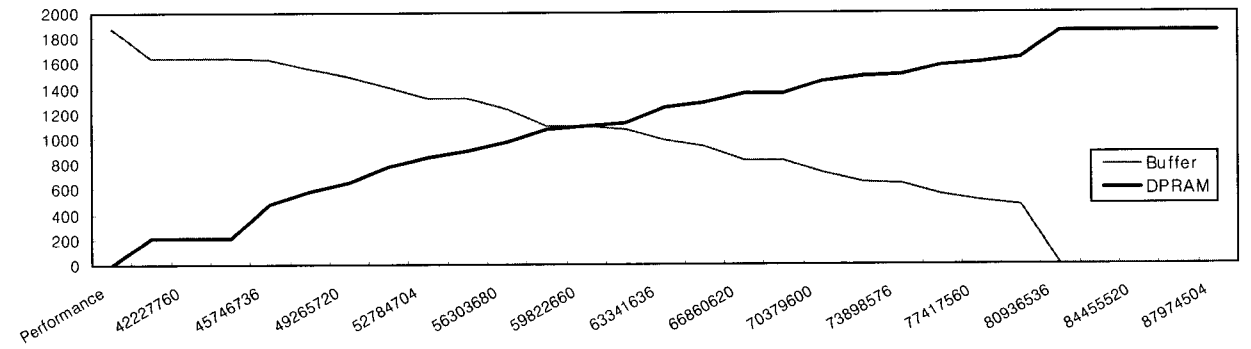
문제점을 보이기 위해 시뮬레이션을 설계하면 다음과 같다. 실험에서 사용하는 입력 데이터의 모델은 입력대역폭이 1518Byte-40Mbps, 최대 Burstiness이며, 부 프로세서 시스템은 성능을 고정시킨다. 이 시뮬레이션에서 변수로서 주 프로세서의 성능을 변경시킨다. 주 프로세서의 처리 성능을 향상시키기 위해서 시스템 클럭을 상승시켜 주 프로세서의 데이터 처리량이 각각 50Mbps, 70Mbps가 되도록 한다. 이러한 상황에서 시뮬레이션 결과를 산출하면 (그림 14)에서 나타나는 것과 같은 반응을 보인다.

일단 시스템의 성능이 입력 대역폭보다 크며, 부 프로세서의 성능이 주 프로세서보다 낮으므로 시스템은 안정하고, 메모리는 일정량을 사용하게 된다. 위 두 가지의 주 프로세서 성능(50, 70Mbps 처리)에 대한 메모리 사용량을 분석하면, 50Mbps 처리쪽(a)은 버퍼 메모리의 사용량(그림의 굵은 선쪽)이 많은 반면, 70Mbps 처리쪽(b)은 공유 메모리의사용량이 많다. 이는 메모리 사용량을 제한하지 않은 상황에서 주 프로세서의 성능이 향상되어 버퍼 메모리로부터 공유 메모리로의 데이터 전달이 빨라지게 되었으며, 그 결과 버퍼 메모리의 사용량이 줄고 공유 메모리의 사용량이 증가 하였음을 나타낸다. 즉, 주 프로세서의 성능을 향상시킨 결과가 오히려 비싼 메모리를 많이 사용하게 되어 경제성에 어긋나게 만든 것이 된다.

그러므로 공유 메모리를 제한하여 주 프로세서 시스템에서 공유 메모리의 사용을 기다리도록 하는(이 때 사용 허가를 기다림으로 인한 평균처리속도가 저하됨) 메모리 제한



(a) 64byte-11Mbps Input



(b) 1518byte-40Mbps Input

(그림 15) 버퍼 및 공유 메모리 사용량 관계

기법을 도입하여야 경제적인 설계를 할 수 있다. 주 프로세서 시스템에서는 공유 메모리가 제한량까지 차게 되면 버퍼 메모리에서 공유 메모리로의 데이터 이동을 금지시키게 되는데, 이에 따라 버퍼 메모리 사용이 증가하게 되어 공유 메모리 제한의 효과가 나타나게 된다. 이러한 제한 작용을 통해 시스템의 안정성이 저하되지 않는 범위 내에서 원하는 메모리 사용 비율을 결정할 수 있게 된다. 두 메모리의 사용량에 대한 관계 시뮬레이션의 결과가 (그림 15)(x축-주 프로세서 성능, y축-버퍼 사용량, 항목-버퍼, 공유 메모리)와 같이 나타난다.

64byte입력을 시뮬레이션한 (a)를 보면, 주 프로세서 시스템이 부 프로세서 시스템보다 낮은 성능을 보이는 그래프의 첫 부분에서는 공유 메모리의 사용량이 2로 되어 있다가 성능이 역전되는 순간부터 공유 메모리 사용량이 증가하고 버퍼 메모리의 사용량이 감소하는 모습을 보이게 된다. 이러한 분포는 1518Byte를 시뮬레이션한 (b)에서도 같은 형태로 나타난다. 그래프에서 나타나는 것과 같이 경제성을 고려하여 공유 메모리의 사용량을 제한하려고 한다면 두 메모리(버퍼, 공유)의 사용량 관계에 의해 시스템에 안정한 버퍼 메모리의 크기를 결정할 수 있게 된다.

시뮬레이션 결과를 토대로 해당 대역폭 내에서 손실이 발생하지 않는 메모리 사용량을 산출하면 다음과 같다. 입력 데이터 크기에 대해 각 대역폭의 해당 burstiness가 설정되면 이 burstiness 동안의 입력 패킷 수에서 burst한 전달 시간 동안 주 프로세서에 의해 버퍼 메모리로부터 공유 메모리로 이동된 양을 뺀 만큼이 버퍼 메모리의 최대 사용량이 되며, 공유 메모리의 사용량은 버퍼 메모리에 데이터가 존재하는 시간 동안 주부 프로세서의 성능 차이 만큼의 양이 최대 사용량이 된다.

4. 실험 및 분석

실제 다중 프로세서 시스템을 이용하여 위의 이론, 시뮬레이션에서 산출한 데이터와 실제 데이터를 비교해 볼 수 있다. 실제 실험은 네트웍 프로세서를 이용하여 설계된 1 : 4 다중 프로세서, 데이터 전송을 위해 사용한 Dual-Port 메모리를 기준으로 시뮬레이션에서와 같이 이더넷 기반에서 실험되었다.

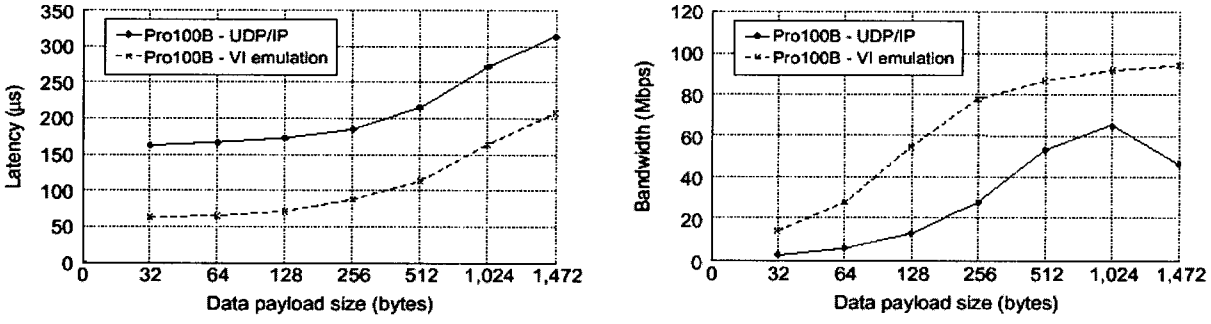
4.1 실험 환경

실제 실험을 위해 필요한 실제 구성 요소들과 테스트 환경 등에 대해 설명한다.

네트웍 실험에서의 입력 모델 조건은 필요로 하는 대역폭을 낼 수 있느냐와 원하는 형태의 패킷(크기, 타입 등)을 만들어 낼 수 있느냐에 달려있다. 이러한 용도를 위해 PC 기반에서 동작하는 몇몇 프로그램들이 있지만, PC의 구조상 출력 대역폭이 작거나 미세한 대역폭 조정이 불가능한 단점이 있다. (그림 16)은 IEEE Micro(1998)에서 보여준 결과치로 dual 200-MHz Pentium Pro 프로세서, 64 Mbytes 메모리, Intel Ether-Express Pro/100B 네트웍 카드에 응용 프로그램을 동작시켜 네트웍 패킷(UDP)의 출력을 실험한 것이다. 패킷의 크기가 작은 경우 출력 대역폭이 작고, 전체적으로 비 선형적이어서 대역폭 조정이 쉽지 않음을 알 수 있다.

이에 실험의 입력 모델로써 패킷의 생성과 흐름의 세부 제어가 가능한 하드웨어 기반 패킷 생성기(Packet Generator)를 도입하였다. 이는 기기 내부의 메모리에 모델링 하고자 하는 패킷을 미리 생성하여 출력할 수 있도록 설계된 것으로, 여러 네트웍 계층(Network Layer)을 거치며 느려질 수 있는 요인이 배제되고, 패킷 간의 전송 지연 조정도 외부 요인의 영향을 받지 않는 하드웨어 타이머를 사용하므로 정확한 형태의 대역폭 생성이 가능해 진다. 또, 전송할 실험용 패킷은 응답을 원하지 않아 추가적인 트래픽을 발생시키지 않는 UDP 패킷으로 하였다.

한편, 시스템에서 출력되는 패킷의 손실이나 대역폭의 측정을 위해서 망을 모니터링할 필요가 있다. 망을 모니터링하는 장치 역시 하드웨어 기반의 장치와 PC 기반의 프로그램이 존재하는데 둘 다 장단점을 가지고 있다. PC 기반의 프로그램에서는 PC 구조로 인해 완전한 네트웍 대역폭을 다 수용할 수 없지만 일정한도 이내의 대역폭에 대해서는



(그림 16) PC기반 네트워크 인터페이스의 성능([9]참조)

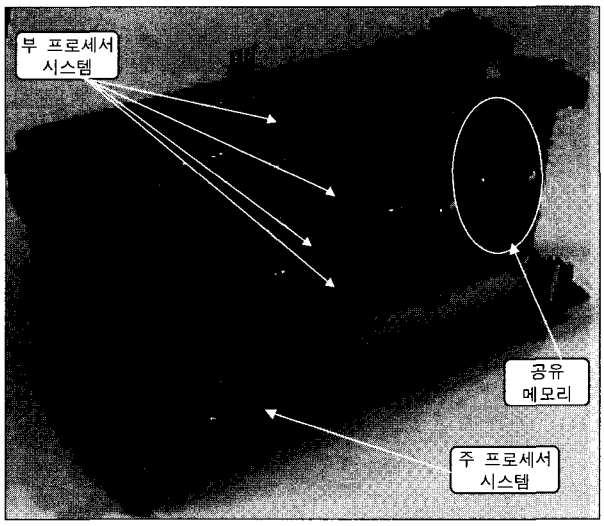
정확한 분석을 할 수 있으므로 PC용 프로그램을 이용하도록 한다.

PC 상에서 사용할 패킷 분석 프로그램으로는 sniffer 사의 Sniffer Pro를 사용하였다. 이 패킷 분석기는 패킷 포획 및 망의 흐름에 대한 통계와 분석 자료를 실시간으로 제공하는 장점으로 인해 네트웍 연구에 많이 사용되고 있다.

〈표 3〉 시스템 제원

구성 항목	주 시스템	부 시스템	특징
CPU	MPC860T	S3C4530A	Motorola 사, Samsung 사
SDRAM	32Mbyte	16MByte	Samsung 사, 32Bit 버스
Ethernet Controller	LXT972A		Intel 사, MII 지원, 10/100Mpbs AutoNego.
DPRAM	Cy7057a		Cypress사, 32bit 버스

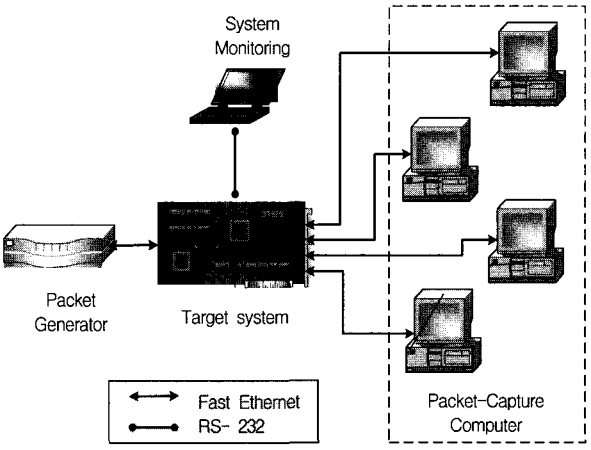
실험을 위해 설계된 다중 프로세서 모델은 이더넷 컨트롤러가 내장된 프로세서이고, 고속 메모리인 SDRAM을 시스템 메모리로, Dual-Port Memory를 공유 메모리로 사용하여 구성한 시스템이다. 주 프로세서로는 모토롤라사의 PowerPC 코어 기반의 MPC860T이고, 부 프로세서들은 ARM7TDMI 코어 기반의 삼성 NetARM S3C4530이다. 고속 이더넷 연결은 100Mbps 인터페이스 표준인 MII(Media Independent Interface)를 사용하였다.



(그림 17) 실제 다중 프로세서 시스템 모델

4.2 실험 내용

앞 절에서 설명한 자원들을 이용하여 (그림 18)과 같은 실험 환경을 구성하였다. 테스트할 다중 프로세서 시스템의 콘솔(Console ; RS-232 Null Modem)은 시스템의 자원 사용량을 제한할 수 있는 설정 메뉴를 보여주거나 시스템의 동작 상황을 모니터링하기 위해 사용한다. 입력 단에 연결된 패킷 생성기에는 정확한 입력 대역폭과 burstiness를 지속적으로 만들어 내도록 설정할 수 있으며, 출력 단에서는 패킷 분석기를 이용하여 출력량을 산출하도록 한다. CSMA/CD 망의 특성인 충돌이 발생되지 않으므로 이로 인한 시스템 성능 측정 오류는 발생되지 않는다.



(그림 18) 실험 환경

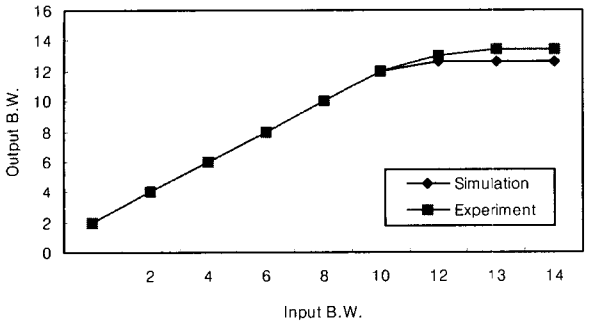
4.3 실험 결과 및 분석

시스템의 각종 설정 변수는 앞장의 시뮬레이션에서 사용한 것과 같다. 실험 결과, 시스템의 실제 처리 능력은 계산치보다 약간 높은 수준에서 측정되었다. 그러나 실험에서도 역시 이 수치 이상의 입력 대역폭에 대해서는 한계치와의 차이만큼 손실되고 있음을 알 수 있다. (그림 19)는 최소, 최대 패킷 크기에 대한 입력 대역폭 중 No Burstiness 상황을 측정한 것이다. 이에 의해 모델 설정에서 이용한 시스템 접근 시간 기법을 이용하여 처리 능력을 산출하는 방식이 적정함을 알 수 있다.

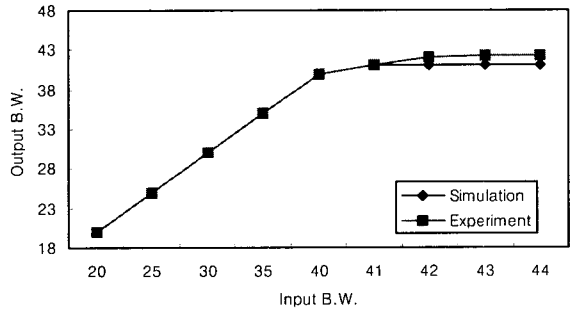
실험에서 처리 능력이 계산치보다 약간 높게 측정된 것은 프로세서 수행 시간과 동기화 작업에 수행되는 명령어 개수의 계산에 약간의 오차가 있을 수 있음과 전처리 단에 사용한 PowerPC 코어의 특징에 기인하고 있다. 접근 시간을 계산할 때 처음으로 사용된, I/O 장치에서 DMA를 이용하여 버퍼 메모리로 데이터를 전송하는 시간(식 (1)) 동안 PowerPC의 코어는 명령어 처리를 병렬로 처리할 수 있게 되는데, DMA가 버스를 사용하는 동안에도 CPU는 Load/Store Unit의 큐(Queue)를 이용할 수 있어 이 시간은 계산에서 제외해도 되므로 성능이 약간 향상될 여지가 있다.

한편, 앞장의 시뮬레이션에서와 같이 입력 대역폭을 11Mbps-64Byte, 40Mbps-1518byte(각각 최대 burstiness)로 하여 데이터 처리 실험을 한 결과 (그림 20)과 같이 64Byte에서는 burstiness에 관계없이 필요한 최대 공유 메모리 사용량이 2로 일정하였으며, burstiness에 따른 영향은 버퍼 메모리에만 나타났으며, burstiness에 대해 대체적으로 선형 증가하였다. 1518Byte에서는 공유 메모리와 버퍼 메모리의

사용량이 모두 2 이상을 넘는 수치로 나타났는데, 이는 이러한 데이터에 대하여 주 프로세서의 성능이 부 프로세서보다 뛰어났기 때문으로 시뮬레이션 결과와 동일하다.

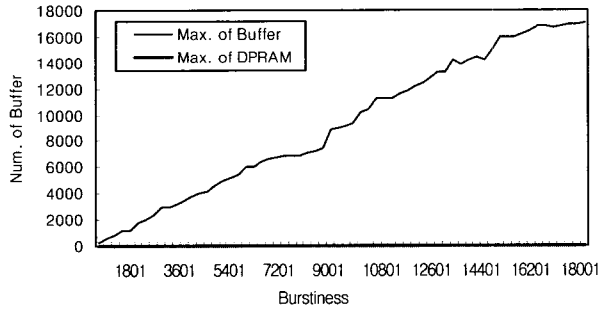


(a) 64Byte-no Burst Input

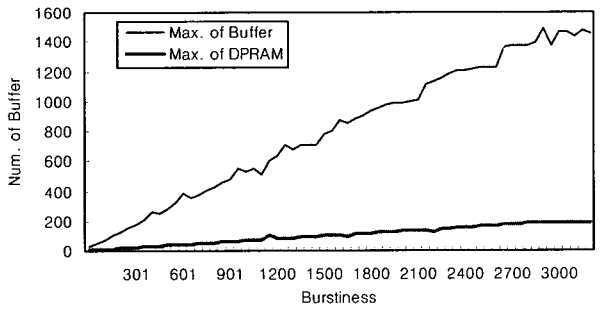


(b) 1518byte-No Burst Input

(그림 19) 시뮬레이션과 실험치의 처리량 비교



(a) 64Byte-Memory amount used



(b) 1518Byte Memory amount used

(그림 20) 실제 버퍼 및 공유 메모리 사용량

공유 메모리로 사용한 Dual-port RAM은 현재 단일 상용 모델로는 1~2Mbyte의 크기가 가장 큰 메모리에 속하며 가격면에서도 상당히 높게 책정되어 있다. 현 시스템에서는 이 메모리의 크기가 64KByte×32로 되어 있기 때문에 내부적으로 메모리 사용이 제한되고 있다. 이러한 상황에서의 실험에서 보인 결과치는 시뮬레이션에서 보이는 버퍼 메모리 대 공유 메모리의 사용량 결과치에 상당 부분 근접하고 있다.(시뮬레이션에서는 실제 실험 결과 보다 약간 기울기가 작은 직선임)

이러한 실험 결과들에 의해 시뮬레이션 상에서의 분석이 실제 실험과 유사한 결과를 나타내고 있음을 확인할 수 있으며, 이에 따라 시뮬레이션을 통한 모델 검증이 유효함을 알 수 있다. 그러나 실제 실험에서 사용되는 코드(수행 명령어)의 개수를 정확히 파악하는 작업에 있어서는 일반적인 프로세서의 성능을 측정하는 방식[6]과 달리 짜여진 코드에서의 명령어 수행 개수를 산출하게 되므로 오차가 발생할 수 있는 가능성을 배제할 수 없으며, 이를 감안하여야 할 것이다.

5. 결론 및 향후 과제

본 논문에서는 Loosely Coupled 다중 프로세서 시스템에서 프로세서들간의 통신을 위한 목적으로 사용되는 공유 메모리의 사용량 최적화에 중점을 두어 연구되었다. 입력되는 데이터들이 메모리 부족으로 손실되지 않는 범위 내에서 버퍼 메모리와 공유 메모리의 사용량을 예측하고, 두 메모리 사이의 상대적 사용량 관계를 규명하기 위해 입력 대역폭 제한을 위한 시스템의 접근 시간 계산 방법을 도입하고, 검증을 위해 PC 시뮬레이션과 실제 실험을 수행하였다.

그 결과, 실제 실험 결과와 공유 메모리량을 제한한 시뮬레이션 결과가 약간의 오차만을 보이며 같은 추세를 보였는데, 주 프로세서의 성능이 부 프로세서 성능의 합보다 더 높고 burstiness의 수치가 높아야 공유 메모리의 사용량이 증가한다는 것과 공유 메모리의 사용량 제한이 버퍼 메모리의 증가에 영향을 주며 이들간의 상대적 사용량이 측정될 수 있음을 보였다.

이를 통해 시스템 내에서의 데이터 무손실을 보장하기 위한 입력 대역폭의 제한 지점과 입력데이터의 밀집정도(burstiness)에 따른 버퍼 및 공유 메모리의 사용량 예측이 가능함을 보였고, 또한 경제적인 시스템 설계를 위해 공유 메모리의 사용량을 제한하고, 이에 의해 늘어나는 버퍼 메모리량이 반비례 관계에 있음도 규명하였다.

현재 본 논문에서는 제한된 데이터 입력 모델(최소, 최대

6) 간단한 공식으로 정확한 명령어 개수를 산출하고 이를 반복 계산하는 방식

의 고정 패킷 크기의 이상적에서 최악의 상황까지만을 설정)만을 사용하였으나 실제 망에서와 같은 가변 패킷 길이와 burstiness를 갖는 입력 모델을 대상으로 한 연구도 진행되어야 할 것이며, 또한 다중 프로세서 시스템의 일반적 모델 설정을 위해 필요한 다른 변수들에 대해서도 연구되어야 할 것이다.

참 고 문 헌

[1] M. Morris Mano, "Computer System Architecture," 3rd Ed. Prentice-Hall, pp.489-512, 1993.

[2] Sundar Iyer, "Analysis of a Memory Architecture for Fast Packet Buffers, : 2001 IEEE Workshop on High Performance Switching and Routing, May, 2001.

[3] G. Gogniat, M. Auguin, "Communication synthesis and HW/SW integration for Embedded System Design," CODES/CASHE'98, 1998.

[4] Juan Carlos Gomez, "The CLAM Approach to Multithreaded Communication on Shared-Memory Multiprocessor : Design and Experiments," IEEE Transactions on Parallel and Distributed Systems, Vol.9, pp.36-49, January, 1998.

[5] Motorola Inc., "860T Design Advisory 0.3," www.mot-sps.com.

[6] Samsung Inc., "S3C4510B RISC Microcontroller," www.samsungsemi.com.

[7] Cypress, "Understanding Asynchronous Dual-Port RAMs," www.cypress.com/whitepaper/, 1997.

[8] Raoul A. F. Bhoedjang "User level network interface protocols," IEEE Computer, pp.53-61, November, 1998.

[9] Dave Dunning, Greg Regnier, "The Virtual Interface," IEEE Micro, pp.66-76, March/April, 1998.

[10] Janaki Akella, Daniel P. Siewiorek, "Modeling and Measurement of the Impact of Input/Output on System Performance," Computer Architecture Conference Proceeding, pp. 390-399, 1992.

[11] Zheng Wang, Jon Crowcroft, "Analysis of Burstiness and Jitter in Multimedia Communications," ACM SIGCOMM Symp., pp.13-19, 1993.

[12] S. Fong, S. Singh, "Modeling and Performance Analysis of Shared Buffer ATM Switches with HotSpot Pushout under Bursty Traffic," Australian Telecommunication and Network Application Conference, Vol.2, pp.561-566, Dec., 1996.

[13] Amr A. Awadallah, Chetan Rai, "TCP-BFA : Buffer Fill Avoidance," IFIP High Performance Networking Conference, September, 1998.

[14] B. Ryu and S. Lowen, "Fractal Traffic Models for Internet Simulation," IEEE ISCC, July, 2000.

[15] 최건, 민상렬, 김종상, "공유 메모리 다중프로세서의 동기를 위한 효율적 메모리 접근 기법", 한국정보과학회논문지 Vol. 20, pp.1377-1390, 1993.

[16] 김영신, 권욱현, "공유 메모리를 통한 신뢰성 있는 데이터 교환 프로토콜", 전자공학회논문집, 제21권 제1호, pp.251-254, 1998.

[17] "공유 메모리와 단일 버스로 구성되는 다중 프로세서의 하드웨어 성능분석", 한국정보과학회논문지 Vol.16, pp.399-409, 1989.

김 종 수

e-mail : promise@ajou.ac.kr

2000년 아주대학교 전자공학과(학사)

2002년 아주대학교 전자공학과(석사)

2002년~현재 아주대학교 전자공학과 박사과정

관심분야 : 초고속 통신망, 임베디드 시스템, 정보보안, RTOS 등

문 종 욱

e-mail : lache@ajou.ac.kr

2000년 아주대학교 전자공학과(학사)

2002년 아주대학교 전자공학과(석사)

2002년~현재 아주대학교 전자공학과 박사과정

관심분야 : 네트워크 보안, RTOS, 임베디드 시스템 등

임 강 빈

e-mail : hotspot@csl.ajou.ac.kr

1992년 아주대학교 전자공학과(학사)

1994년 아주대학교 전자공학과(석사)

2001년 아주대학교 전자공학과(박사)

1999년~2000년 (미)아리조나 주립대 객원 연구원

1997년~현재 아주대학교 정보통신전문대학원 전임연구원

관심분야 : 실시간 운영체제, 네트워크 보안, 내장형 시스템 등

정 기 현

e-mail : khchung@ajou.ac.kr

1984년 서강대학교 전자공학과(학사)

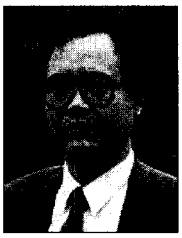
1988년 Univ. of Illinois, EECS(석사)

1990년 Univ. of Purdue, 전기전자공학부 (박사)

1991년~1992년 현대반도체 연구소

1993년~현재 아주대학교 전자공학부 교수

관심분야 : 컴퓨터 구조, VLSI 설계, 멀티미디어 및 실시간 시스템 등



최 경 희

e-mail : khchoi@ajou.ac.kr

1976년 서울대학교 사범대학 수학교육과 (학사)

1979년 프랑스 그랑데꼴 ENSEIHT, 정보공학 및 응용수학(석사)

1982년 프랑스 Univ. of Paul Sabatier (박사)

1991년~1991년 프랑스 렌느 IRISA 연구소 교환 교수

1982년~현재 아주대학교 정보 및 컴퓨터 공학부 교수

관심분야 : 운영체제, 분산처리, 실시간 시스템 등