論文94-31A-6-21

멀티프로그래밍 환경에서의 새로운 DRAM 구조의 성능 분석

(Performance Evaluation of the New DRAM Architectures in Multiprogramming Environment)

安 台 源*, 鄭 德 均*, 민 상 렬***, 최 윤 호**

(Tae Won Ahn, Deog Kyoon Jeong, Sang Lyul Min and Yoon-Ho Choi)

要 約

CPU와 DRAM 간의 속도차는 현재 컴퓨터 시스템의 성능 향상에 큰 장애가 되고 있으며 시스템 설계에서 큰 문제를 제기하고 있다. 이러한 문제점을 적은 비용으로 해결하기 위하여 기존의 DRAM 구조를 변형시킨 새로운 DRAM 구조들이 제안되어 왔다. 이 논문에서는 몇 가지 새로운 DRAM 구조가 전체 시스템의 성능에 미치는 영향을 정량적으로 평가하기 위하여 캐쉬 및 메모리 시뮬레이터를 개발하였고 4가지 benchmark 프로그램의 수행 중에 발생한 트레이스를 바탕으로 모의 실험을 하였다. 새로운 DRAM 구조는 기존의 DRAM으로 구성된 경우에 비하여 저가격 시스템의 경우 5~30% 정도, 고성능 시스템의 경우에는 1% 미만의 성능 향상이 기대됨을 알 수 있었고 보다 실제적인 환경인 멀티프로그래밍의 효과를 고려하면 저가격 시스템의 경우 1~3%, 고성능 시스템의 경우에는 약 10~28% 정도의 추가된 성능 향상이 기대됨을 알 수 있었다.

Abstract

In the design of modern computer systems, the speed gap between the CPUs and DRAMs has been a major concern. To relieve this problem at a low cost, several new DRAM architectures have been proposed. This study is aimed at evaluating quantitatively the impact of the new DRAM architectures (synchronous DRAM, dual-RAS synchronous DRAM, and enhanced DRAM) on the memory system performance. We developed a cache and memory simulator and performed various experiments using the traces generated from four benchmark programs. The simulation results show that the new DRAM architectures offer a better performance than a conventional one by 5~30% in a low cost system and their improvement in a high performance system is less than 1%. However, for reasonable multiprogramming workloads, additional performance improvement of about 10~28% is expected in a high performance system while 1~3% in a low cost system.

* 正會員, 서울大學校 電子工學科
(Dept. of Elec. Eng., Seoul Nat'l Univ.)
** 正會員, 三星電子 半導體研究所
(SAMSUNG, Electronic Semiconductor Business)
*** 正會員, 서울大學校 컴퓨터工學科
(Dept. of Computer Eng., Seoul Nat'l Univ.)
接受日字 : 1993年 9月 1日

Ⅰ. 서론

현재 컴퓨터 시스템에 사용되는 메모리의 주종을 이루는 DRAM의 속도는 고속의 프로세서와 비교할 때 상대적으로 매우 느리며 이미 그 속도의 한계성을 극복하기 위하여 기존의 DRAM 구조를 변형시킨 새로운 구조의 DRAM 개발이 추진되었다. [1]

새로운 구조의 DRAM은 가격이나 성능 면에서 기존의 DRAM과 비교했을 때 경쟁력을 갖는 것이 무엇보다도 중요하므로 정확한 성능 분석을 위해서는 DRAM이 사용되는 메모리 시스템에 대한 전체적인 이해가 필요하다.

DRAM의 성능은 주소가 주어진 때부터 최초의 데이타가 전달되는 시간인 접근응답시간 (latency), 데이타의 최대 전송 속도인 대역폭 (bandwidth) 등으로 평가되지만 일반적으로 계층적 구조를 갖는 메모리 시스템의 경우 전체 시스템의 성능은 각 단계 간의 상호 작용으로 말미암아 어느 한쪽만으로 그 성능을 평가하기는 어려우며 시스템 동작시의 평균 접근시간 (average access time)으로밖에 나타낼 수 없다. 이 값은 실제 응용 프로그램이 수행될 때 발생되는 메모리 접근 형태 (memory access pattern)인 트레이스 (trace)를 바탕으로 한 모의 실험을 통하여 얻어낼 수 있다. [2,3]

본 논문에서는 이러한 관점에서 컴퓨터 시스템을 저가격 시스템 (low cost system)과 고성능 시스템 (high performance system)으로 분류한 후 각각의 경우에 대하여 새로운 DRAM 구조에 의한 메모리 시스템의 성능 향상을 정량적으로 비교하여 보고자 한다

Ⅱ. 메모리 시스템의 계층적 구조

현재 대부분의 컴퓨터 시스템에서는 프로세서와 메모리 간의 속도차에 의한 시간 지연을 줄이기 위하여 그림 1에서와 같이 메모리 구조를 계층적으로 구성하는 것이 일반화되었다. 이러한 메모리 구조의 최상위 단계는 1-단계 캐쉬 (1st-level cache)로서 작은 크기의 매우 빠른 SRAM으로 만들어지고 최하위에는 DRAM으로 구성된 주메모리 (main memory)가 있다. [4-6]

캐쉬는 프로세서로부터 고속으로 접근될 수 있는 메모리로서, 자주 참조되는 메모리 블럭을 저장하여 이에 대한 접근 시간을 줄이는 역할을 한다. 이것은 프로세서와 메모리 간의 속도 차이에서 오는 성능의 저하를 막는데 매우 효과적이다. 그러나, 프로세서와 메모리

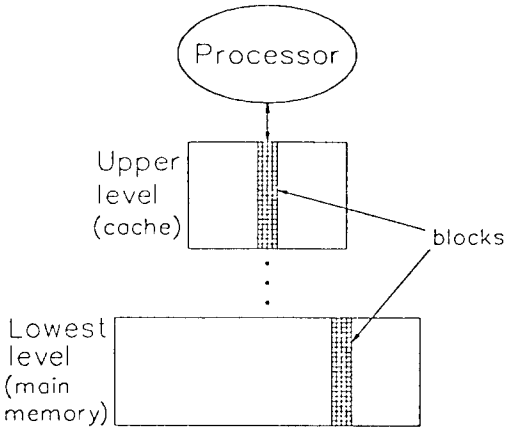


그림 1. 계층적 메모리 구조
Fig. 1. Memory hierarchy.

간의 속도 차이가 커짐에 따라 캐쉬 접근실패 (cache miss)가 시스템의 성능에 미치는 영향은 점점 더 커지고 있다. 따라서 최근의 고성능 컴퓨터 시스템에서는 비교적 큰 크기의 외부 캐쉬를 추가하여 캐쉬를 다단으로 구성함으로써 1-단계 캐쉬에서의 접근실패 (miss)에 의한 긴 시간 지연을 줄이도록 하고 있다. [7]

캐쉬에서 접근실패가 일어나면 그 다음 단계의 메모리 블럭에 대한 접근이 일어나는데 이 때 각 단계를 연결하고 있는 버스의 크기, 전송되는 블럭의 크기 및 전송 가능 속도 등에 따라 지연되는 시간이 정해지게 된다. 캐쉬에서의 접근실패가 상대적으로 매우 느린 DRAM으로 구성된 주메모리까지 이르게 되는 빈도를 줄이기 위해서는 캐쉬 접근실패를 최소로 하여야 하는데 이것은 캐쉬의 크기, 접근성공/접근실패의 최소 기본 단위인 캐쉬 블럭 크기 및 하나의 블럭이 저장될 수 있는 위치의 갯수인 연관도 (set-associativity) 등의 캐쉬 인자 (cache parameters)에 의해서 결정된다. [5,6] 따라서 메모리 시스템의 설계에는 CPU가 메모리를 접근하는 주소 형태, 버스의 크기, 데이타의 전송 속도, 그리고 시스템의 성능을 높이기 위해 추가되는 비용 등에 관한 여러 가지 정보가 필요하다.

Ⅲ. 새로운 DRAM 구조의 특징

이번 모의 실험에서 기존의 DRAM과 성능이 비교된 새로운 구조의 DRAM의 특징을 간단히 요약하면 다음과 같다. [1]

1. Synchronous DRAM (동기식 DRAM)

비동기적으로 동작하는 기존의 DRAM과 비교하여 synchronous DRAM이 갖는 특징은 모든 데이타를 DRAM에 공급되는 클락에 동기시켜 읽고 쓴다는 것이다. 이것은 주소가 주어진 후 데이타를 취할 수 있는 시점을 클락과 동기시켜 미리 정확히 알 수 있기 때문에 특히 연속적으로 데이타를 전송할 때 빠른 속도를 낼 수 있다. 따라서 DRAM을 접근하는 방법은 기존의 DRAM과 비슷하나 최대 전송 가능 대역폭을 증가시킨 것이 이 구조의 특징이며 내부의 구조는 변하지 않고 I/O 기능만 향상시켰으므로 기존의 DRAM에 비하여 접근응답시간 (latency)의 향상을 기대할 수는 없다.

2. Dual-RAS synchronous DRAM (이중-RAS 동기식 DRAM)

서로 다른 행 주소의 데이타가 DRAM 내에서 서로 다른 뱅크에 저장된다면 여러 개의 페이지가 동시에 선택된 상태가 되므로 각 뱅크 당 RAS (Row Address Strobe) 신호를 따로 갖게 되면 페이지 접근성공 (page hit)이 일어날 확률이 늘어난다. 따라서 여러 뱅크 중 한개의 페이지 접근성공이 난 경우 precharge 시간과 RAS 접근 시간을 줄일 수 있다. 이번 논문에서는 synchronous DRAM의 RAS 신호를 2 개로 한 구조에 대하여 모의 실험을 수행하였다.

3. Enhanced DRAM (캐쉬-내장 DRAM)

DRAM chip 내에 SRAM 캐쉬를 포함시킨 구조로서 DRAM array와 on-chip SRAM 캐쉬 사이는 넓은 대역폭을 갖는 버스로 연결되어 있다. 또한 chip 내에 캐쉬 접근성공/접근실패를 알아내는 비교기와 캐쉬 로우 레지스터 (cache row register)를 두어 precharge에 의한 시간 지연을 없애고 있다. 이번 논문에서는 1Mx4 DRAM chip 내에 4개의 512 byte 캐쉬 블럭을 내장시킴으로써 4개의 행주소에 대하여 CAS (Column Address Strobe) 신호에 의한 빠른 접근이 가능한 구조에 대한 모의 실험을 수행하였다.

Ⅳ. DRAM 구조의 성능 평가 방법

1. 모의 실험 방법

모의 실험 방법은 그림 2와 같다. 먼저 benchmark 프로그램을 실행시켜 명령어 추적기 (instruction tracer)로부터 프로그램의 트레이스를 얻는다. 트레이스에는 명령어, 데이타 읽기, 데이타 쓰기를 나타내는 표시 (flag)와 각각에 해당되는 주소 정보가 담

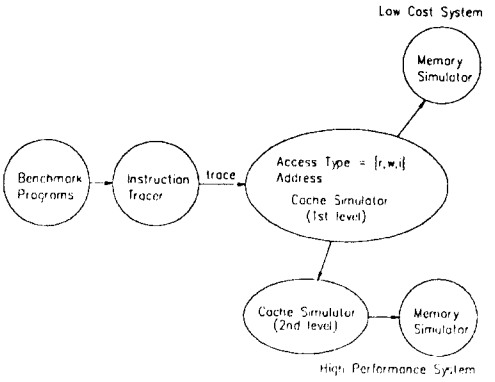


그림 2. 모의 실험 방법
Fig. 2. Simulation Method.

겨 있다. 발생된 트레이스는 즉시 (on-the-fly) 1-단계 캐쉬에서의 모의 실험을 위하여 캐쉬 시뮬레이터로 보내진다.

캐쉬 시뮬레이터 (cache simulator)는 명령어 캐쉬, 데이타 캐쉬, 또는 통합 캐쉬 (unified cache)를 모두 처리할 수 있으며 캐쉬 크기, 캐쉬 블럭 크기, 연관도 (set-associativity) 등의 캐쉬 인자 (cache parameter)와 트레이스 형식으로 많이 사용되는 din 입력에 대하여 각각의 캐쉬 접근에 대한 모의 실험을 하게 된다. din 형식은 benchmark 프로그램이 수행되면서 발생한 주소 (address)와 그 주소의 접근 유형 (access type)을 나타내는 표시 (flag)로 구성되는데 다음 표 1과 같이 정의된다.

표 1. 트레이스 데이타 형식
Table 1. Trace data format.

flag	주소의 접근 유형
0	데이타 읽기 (Data Read)
1	데이타 쓰기 (Data Write)
2	명령어 (Instruction)
3	unknown access type
4	캐쉬 플러쉬 (Cache Flush)

따라서 현재 입력된 주소를 flag에 따라 구분하여 캐쉬의 인덱스로 사용될 주소 영역으로 캐쉬 내용 (cache entry)을 찾은 후 태그에 해당하는 주소 영역을 비교하여 일치하면 캐쉬 접근 성공으로 결정하

여 통계를 내게 된다. 캐쉬에서 접근 실패가 일어난 경우에는 다음 단계에서의 모의 실험을 위하여 역시 din 형식의 표준 출력으로 트레이스를 넘겨준다. 만약 flag가 4인 접근이 시도되었다면 즉시 캐쉬의 모든 내용을 무효화시키고 변경된 데이타는 copy-back을 위해 트레이스를 발생시킨다. 이것은 멀티프로그래밍 환경에서 문맥 교환이 일어나서 캐쉬를 플러쉬시키는 경우에 사용한다.

메모리 시뮬레이터 (memory simulator)는 캐쉬에서 발생된 접근실패의 트레이스로부터 여러 가지 DRAM 구조에 대하여 페이지 접근성공/접근실패에 대한 통계를 내기 위한 것으로서 DRAM 구조에 대한 주요 인자로는 주메모리의 뱅크를 결정하는 주소 영역과 DRAM chip의 행주소, 열주소에 사용되는 주소 영역이 있다. 또한 dual-RAS 및 캐쉬-내장 DRAM의 모의 실험을 위하여 행주소를 기억할 수 있는 갯수와 DRAM chip 내부 뱅크의 갯수를 지정할 수 있도록 하였다.

따라서 4 개의 뱅크로 이루어진 주메모리를 가정한 경우 DRAM을 접근하는 주소의 상위 3 bit로 뱅크를 선택하는 방법을 사용하여 dual-RAS의 효과를 얻었으며 1Mx4 DRAM chip 내에 4개의 512 byte 캐쉬를 내장한 enhanced DRAM의 경우는 선택된 뱅크의 DRAM chip 내에 4 개의 행주소를 기억하고 DRAM의 페이지에 해당하는 주소 영역 (행주소)을 비교함으로써 페이지 접근성공/접근실패 여부를 판별하도록 하여 그 효과를 모의 실험하였다.

여기서 명령어 추적기로부터 발생된 트레이스는 가상 주소 (virtual address)이므로 이것을 물리 주소 (physical address)로 천이 (translation)시키는 과정이 필요한데 이것은 각각의 가상 주소에 대하여 가상 페이지 번호 (Virtual Page Number, VPN)를 물리 페이지 번호 (Physical Page Number, PPN)로 변환시키는 과정으로 설명될 수 있다. 이번 모의 실험에서는 4 Kbyte의 페이지 크기를 가정하여 32 bit 가상 주소의 상위 20 bit (VPN)에 대하여 난수 (random number)를 발생시켜 물리 페이지 번호 (PPN)로의 천이표 (translation table)을 만들어 가는 방법을 사용하였다.

2. 사용한 benchmark 프로그램

트레이스를 발생시키기 위한 명령어 추적기로는 SPARC 구조의 컴퓨터에서 사용될 수 있는 shadow라는 프로그램을 사용하였고[8] benchmark 프로그램으로는 Boolean 표현으로부터 진리표를 구성하는 eqntott, 핵반응로에서의 열수소화 계산을 하는 doduc, 회로 시뮬레이터인 spice3, 자동 레이아웃 생성 도구 (automatic layout generation tool)인 wolfe가 사용되었다. eqntott, doduc는 10 가지 프로그램으로 구성된 SPEC benchmark 중에서 선택한 표준 benchmark 프로그램으로서 eqntott는 정수형 연산이 위주인 SPECint이고 doduc는 부동소수점 연산이 많은 SPECfp이다. spice3, wolfe는 VLSI 설계용 도구인 OCTTOOLS에서 선택하였고 여러 가지 입력에 대한 프로그램 수행이 가능한데 본 논문에서는 4 bit 덧셈기에 대한 회로 시뮬레이션과 8 bit 에러 정정 논리 회로에 대한 자동 레이아웃 생성을 각각 수행시켰다.

트레이스 구동 모의 실험 (trace driven simulation)에는 큰 디스크 공간과 많은 시간의 노력이 필요하므로 계속적으로 발생되는 명령어 주소 및 데이타 주소의 처리에 대하여 최대한의 효율을 얻기 위한 방법이 필수적이다. 이번 모의 실험에서는 여러 가지 DRAM 구조에 대하여, 그리고 고성능 시스템의 경우에는 여러 가지 2-단계 캐쉬 구조에 대한 실험 결과를 필요로 하므로 캐쉬에서 접근실패가 난 경우에 한하여 트레이스를 만들며 이를 압축하여 저장하는 방법을 사용하여 비교적 적은 디스크 공간에서도 수행이 가능하도록 하였다.

3. 가정한 시스템

그림 3과 같은 두 가지 유형의 시스템에 대하여 모의 실험을 수행하였다.

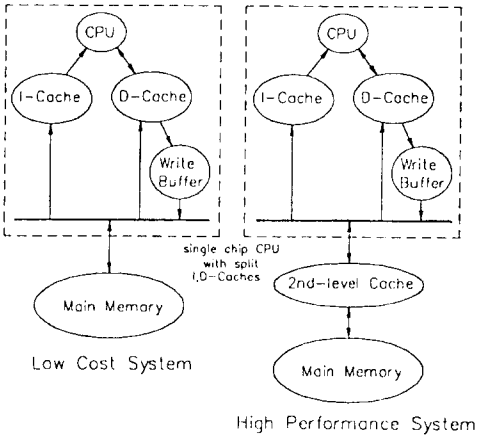


그림 3. 가정한 시스템의 구조
Fig. 3. System configurations.

저가격 시스템은 현재의 마이크로프로세서에서 프로세서와 주메모리 사이에 on-chip 캐쉬만을 가지고

있는 시스템으로 가정하였는데 on-chip 캐쉬는 chip 내부에 집적시키는 면적의 제한으로 인하여 비교적 작은 크기를 갖는 것이 보통이며 본 논문에서는 40 MHz의 CPU가 캐쉬 크기 16 Kbyte, 4-way 연관도, 캐쉬 블럭 크기 32 byte인 명령어 캐쉬, 이와 똑같은 구조의 데이타 캐쉬를 chip 내부에 갖는다고 가정하였다. 고성능 시스템은 저가격 시스템과 동일한 1-단계 on-chip 캐쉬를 가지며 여기에 시스템의 성능을 높이기 위한 2-단계 캐쉬가 CPU 외부에 추가되는데 여러 가지 캐쉬 크기 (256 Kbyte~4 Mbyte), 캐쉬 블럭 크기 (32 byte~128 byte), 연관도 (direct-mapped~4-way set-associativity)에 대하여 각각 모의 실험을 수행하였다.

저가격 시스템은 1-단계 캐쉬와 주메모리 사이에 32 bit 버스가 연결되어 있고 고성능 시스템은 2-단계 캐쉬와 1-단계 캐쉬, 주메모리 사이에 각각 64 bit 버스가 연결되어 있는 것으로 가정하고 1-단계 캐쉬는 접근성공(hit)인 경우 1 클락 주기에 접근이 가능하고 2-단계 캐쉬는 2 클락의 태그(tag) 비교와 4 클락의 전송 (전체적으로 3 클락의 접근응답시간과 3 클락의 전송시간)으로 총 6 클락 주기에 접근이 가능한 것으로 가정하였다 캐쉬에서 접근실패(miss)가 난 경우에는 모두 그 아래 단계의 메모리로 엑세스를 넘기고 따라서 DRAM으로 구성된 주메모리에의 접근이 일어나게 되는데 여러 가지 DRAM에 대하여 페이지 접근성공, 접근실패가 난 경우의 데이타 전송 시간 (HITmm, MISSmm)은 다음과 같이 가정하였다.

저가격 시스템의 경우

저가격 시스템은 CPU 클럭 주파수가 40 MHz이며 1-단계 캐쉬의 블럭 크기가 32 byte이고 32 bit 버스가 연결되어 있으므로 하나의 블럭을 전송하기 위한 DRAM의 CAS (Column Address Strobe) 신호의 반복 (toggling)은 8 번이 된다.

① 기존의 DRAM

HIT_{mm} = 1 (1-단계 캐쉬 접근)
+ 2 (버스 요구와 행주소 비교)
+ 2 × 8 (CAS 신호 반복)
= 19 cycles

$MISS_{mm}$ = 1 (1-단계 캐쉬 접근)
+ 2 (버스 요구와 행주소 비교)
+ 5 (precharge)
+ 5 (RAS 신호)
+ 2 8 (CAS 신호 반복)
= 29 cycles

② Synchronous or Dual-RAS synchronous DRAM

HIT_{mm} = 1 + 2 + 1 8 = 11 cycles
$MISS_{mm}$ = 1 + 2 + 4 + 4 + 1 8 = 19 cycles

③ Enhanced DRAM

HIT_{mm} = 1 + 2 + 1 8 = 11 cycles
$MISS_{mm}$ = 1 + 2 + 4 + 1 8 = 15 cycles

고성능 시스템의 경우

고성능 시스템은 CPU의 동작 주파수가 100MHz이며 저가격 시스템에서와는 달리 하나의 블럭을 전송하기 위한 CAS 신호 반복의 횟수는 2-단계 캐쉬 블럭 크기의 함수가 된다. 이 때 주메모리와 2-단계 캐쉬 사이에는 64 bit 버스가 연결되어 있으므로 CAS 신호 반복의 횟수는 2-단계 캐쉬 블럭 크기 ÷ 8 이 된다.

① Conventional DRAM

HIT_{mm} = 3 (2-단계 캐쉬 접근)
+ 3 (버스 요구와 행주소 비교)
+ 3 (2-단계 캐쉬 블럭 크기 ÷ 8)
(CAS 신호 반복)

$MISS_{mm}$ = 3 (2-단계 캐쉬 접근)
+ 3 (버스 요구와 행주소 비교)
+ 10 (precharge)
+ 10 (RAS 신호)
+ 3 (2-단계 캐쉬 블럭 크기 ÷ 8)
(CAS 신호 반복)

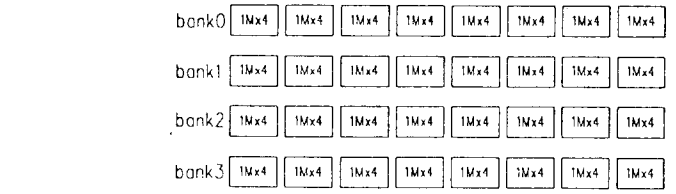
② Synchronous DRAM, dual-RAS synchronous DRAM

HIT_{mm} = 3 + 3 + 1 (2-단계 캐쉬 블럭 크기 ÷ 8)
$MISS_{mm}$ = 3 + 3 + 10 + 10 + 1
(2-단계 캐쉬 블럭 크기 ÷ 8)

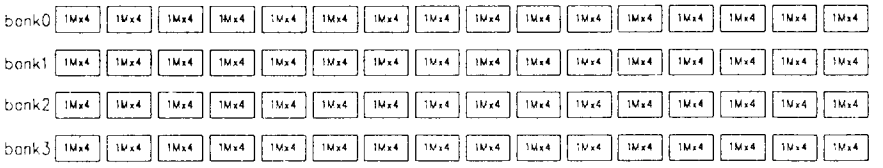
③ Enhanced DRAM

HIT_{mm} = 3 + 3 + 1 (2-단계 캐쉬 블럭 크기 ÷ 8)
$MISS_{mm}$ = 3 + 3 + 10 + 1
(2-단계 캐쉬 블럭 크기 ÷ 8)

이 경우 캐쉬 접근 시간과 버스 요구 및 행주소 비교 시간이 증가하여 실제의 접근응답시간이 커지게 되면 대역폭이 개선된 새로운 DRAM은 기존의 DRAM에 비하여 상대적으로 더 큰 성능 향상을 기대할 수 있게 된다. 또한 두 가지 시스템 모두에 쓰기 버퍼 (write buffer)를 두어 캐쉬에서 변경된 (dirty) 데이타가 주메모리로 쓰여질 때의 시간 손실은 없는 것으로 가정하였다.



(a) 저가격 시스템



(b) 고성능 시스템

그림 4. 가정한 주메모리의 구조
Fig. 4. Main memory configurations.

주메모리는 모두 4 개의 뱅크로 이루어지고 각각의 DRAM chip은 10 bit의 행 주소 (row address)와 10 bit의 열 주소 (column address)를 시간 분할 (time multiplexing)하는 것으로 가정한다면 그림 4에서와 같이 저가격 시스템의 경우 16 Mbyte, 고성능 시스템의 경우 32 Mbyte의 주메모리를 갖게 된다.

V. 결과 및 분석

1. 1-단계 캐쉬 (1st-level cache)의 성능

각각의 benchmark 프로그램이 실행되면서 참조된 명령어, 데이타의 갯수와 1-단계 캐쉬의 성능에 대한 모의 실험 결과는 표 2와 같다. eqntott는 프로그램의 크기가 아주 작으므로 명령어 접근실패율이 매우 낮고, spice3의 경우에는 상대적으로 프로그램의 크기가 크므로 캐쉬 접근실패율이 높은 것이 특징이다.

표 2. 1-단계 캐쉬의 성능
Table 2. Performance of the 1st-level cache.

	eqntott	doduc	spice3	wolfe
실행 프로그램 크기	48 Kbyte	376 Kbyte	4880 Kbyte	487 Kbyte
총 명령어 수	1.780e+09	1.698e+09	2.58e+08	1.551e+08
명령어 접근실패	1294	1.449e+07	1.981e+07	884210
데이타 읽기 수	5.769e+08	4.646e+08	7.130e+07	2.693e+07
읽기 접근실패	1.051e+07	5.531e+06	2.57e+06	999222
데이타 쓰기 수	1.741e+07	1.382e+08	2.46e+07	1.066e+07
쓰기 접근실패	131024	5.298e+06	1.215e+06	166763
되쓰기(Copy-Back)	205023	5.817e+06	1.809e+06	468610
캐쉬 접근실패율	0.5978%	1.4905%	9.1207%	1.3217%

2. 유효 접근 시간

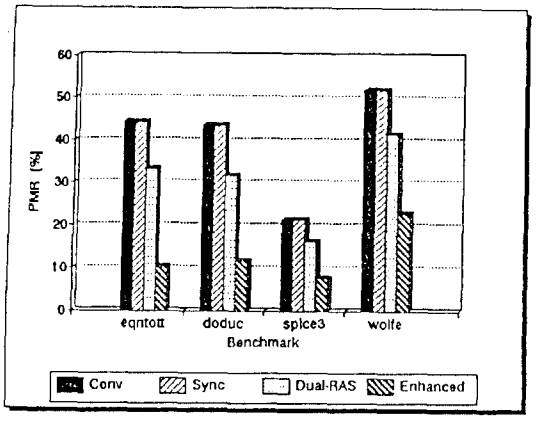
1-단계 캐쉬, 2-단계 캐쉬의 접근실패율 (cache miss ratio)과 주메모리에서의 DRAM 페이지 접근실패율 (page miss ratio, PMR)을 각각 MR1, MR2, MRmm이라 하고 DRAM 구조에 대한 페이지 접근성공, 접근실패에 대한 접근 시간을 각각 HITmm, MISSmm이라고 할 때 메모리에 대한 유효 접근 시간 (effective access time)은 저가격 시스템의 경우 식 (1)로 계산되고 고성능 시스템의 경우에는 식 (2)와 같다. [5]

$$Teff = (1-MR_1) \times 1 + MR_1 \times Teff,mm \quad (1)$$
$$Teff,mm = (1-MRmm) \times HITmm + MRmm \times MISSmm$$

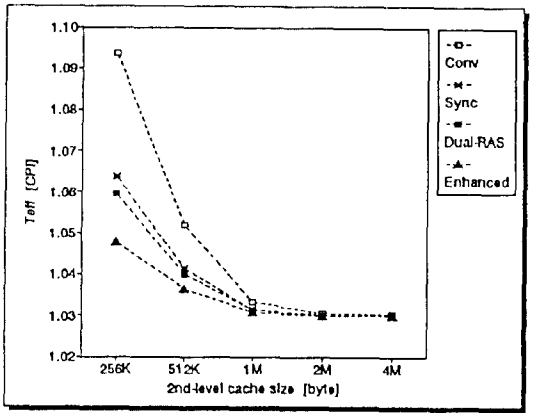
$$Teff = (1-MR_1) \times 1 + MR_1 \times Teff,2nd \quad (2)$$
$$Teff,2nd = (1-MR_2) \times 6 + MR_2 \times Teff,mm$$
$$Teff,mm = (1-MRmm) \times HITmm + MRmm \times MISSmm$$

유효 접근 시간은 평균적으로 한 개의 명령어가 수행되면서 메모리에 접근하는데 필요로 했던 클락 수가 되는데 그 단위는 CPI (Cycles Per Instruction)가 된다.

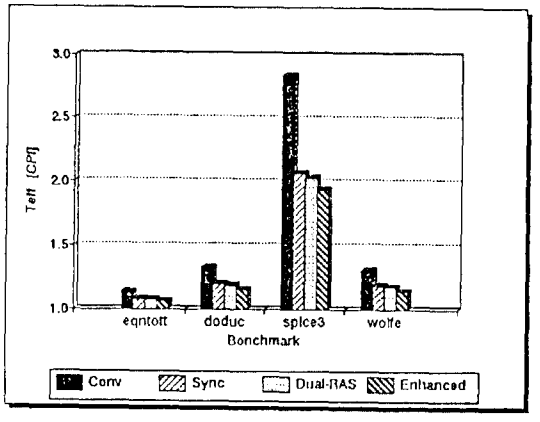
그림 5는 저가격 시스템의 모의 실험 결과이다. spice3의 경우 1-단계 캐쉬에서의 접근실패율이 매우 높아서 DRAM에서의 페이지 접근실패율 (PMR)은 상대적으로 낮음에도 불구하고 Teff는 가장 높은 것



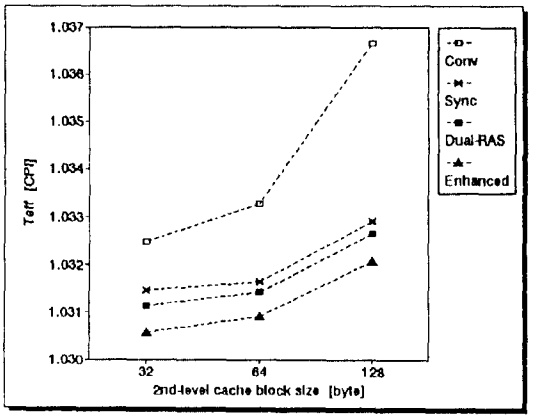
(a) DRAM의 페이지 접근실패율



(b) Teff, 2-단계 캐쉬 크기의 영향

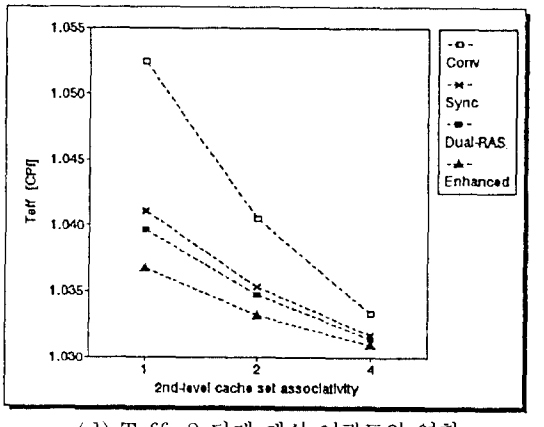


(b) 유효 접근 시간, Teff [CPI]

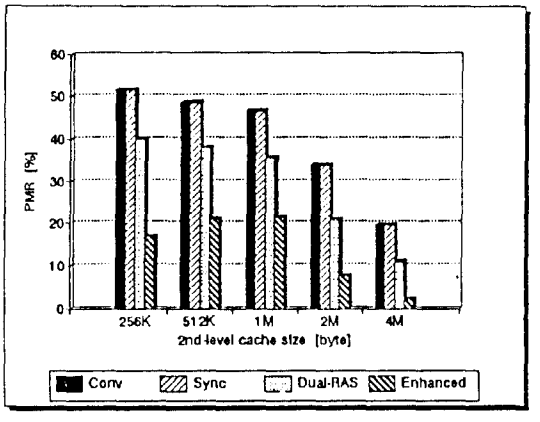


(c) Teff, 2-단계 캐쉬 블럭 크기의 영향

그림 5. 저가격 시스템의 모의 실험 결과
Fig. 5. Simulation results of a low cost system.



(d) Teff, 2-단계 캐쉬 연관도의 영향



(a) DRAM의 페이지 접근실패율

그림 6. 고성능 시스템의 모의 실험 결과 (eqntott)
Fig. 6. Simulation results of a high performance system (eqntott).

으로 나타났다. 이 경우 DRAM의 접근이 자주 일어나므로 DRAM의 성능이 전체 시스템의 성능에 미치는 영향이 크다. DRAM chip 내에 SRAM 캐쉬를 포함하고 있는 enhanced DRAM이 가장 높은 성능 향상을 보였지만 새로운 구조의 DRAM들 간의 차이는 2~3% 정도이며 기존의 DRAM과 비교할 때 eqntott의 경우 5% 정도, doduc와 wolfe의 경우 10% 정도, spice3의 경우 30% 정도의 전체적인 성능 향상을 기대할 수 있었다. Dual-RAS DRAM의 경우는 synchronous DRAM과 비교할 때 성능 차이가 1% 정도로 나타났다.

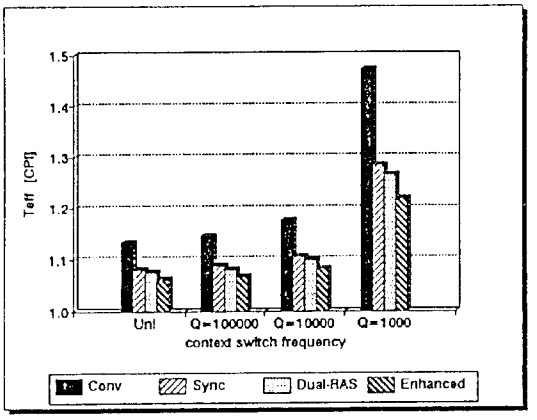
그림 6은 eqntott를 사용한 경우 고성능 시스템에서의 모의 실험 결과를 나타낸다. 그림 6.(a)는 2-단계 캐쉬의 크기의 변화에 따른 캐쉬 접근실패가 DRAM의 페이지 접근실패율에 주는 영향을 보여 준다. 그림에서 보면 2-단계 캐쉬의 크기가 커짐에 따라 DRAM의 페이지 접근실패율이 낮아짐을 알 수 있다. 또한, 2-단계 캐쉬의 크기가 클수록 캐쉬 접근실패가 적어지므로 전체적인 접근 시간은 그림 6.(b)와 같이 되고 2-단계 캐쉬의 연관도가 클수록 역시 캐쉬 접근실패는 적어지고 Teff는 그림 6.(d)와 같은 결과를 보여준다. 한편, 2-단계 캐쉬의 블럭 크기가 커지면 접근실패가 적어지게 되지만 주메모리와의 데이타 전송에 필요한 시간이 늘어나므로 이 두 가지 요소의 상호 작용에 의해 Teff가 결정되는데 그림 6.(c)의 경우는 블럭 크기 증가로 인한 전송 시간의 증가가 캐쉬 접근실패 감소 효과를 능가했기 때문에 나타난 결과이다. 1Mbyte의 2-단계 캐쉬를 사용한 경우 2 % 정도의 낮은 캐쉬 접근실패율로 인하여 새로운 DRAM에 의한 성능 향상은 1 % 미만이다. 그러나 이것은 단일 프로그래밍 (uniprogramming)의 경우이며 멀티프로그래밍의 효과를 고려하면 더 큰 성능 향상을 기대할 수 있는데 다음 절에서 그 결과를 제시한다.

3. 멀티프로그래밍에 의한 효과

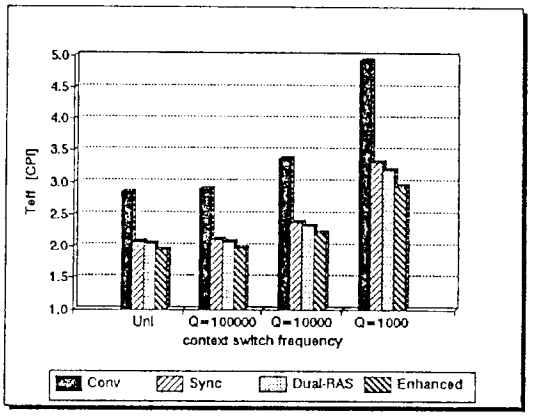
UNIX와 같은 다중 사용자 운영 체제 (Multi-user Operating System)의 경우 CPU는 여러 프로세스 (process)를 시분할 (time-multiplexing)로 처리하게 된다. 이러한 멀티프로그래밍에 의해 프로세스가 바뀌게 되면 (문맥 교환, context switch) 기존의 캐쉬의 내용은 서서히 교체되어 없어지게 된다.[9,10] 통계적으로는 수백 s 간격으로 문맥 교환이 일어난다고 알려져 있으며[6,9] 이것을 모의 실험을 위해서는 캐쉬 시뮬레이터에서 일정한 시간 (명령어) 간격으로 모든 캐쉬 내용을 쓸어냄으로써 (flush) 그 효과를 나타낼 수 있다. 표 3은 40 MHz와 100 MHz의 속도로 동작하는 프로세서에 대하여 문맥 교환이 일어나는 명령어 간격 (Q)에 대한 문맥 교환 시간 간격을 정리한 것이다. 실제로 고성능 시스템의 경우에는 저가격 시스템에 비하여 같은 시간에 더 많은 명령어가 수행되므로 Q가 더 크다고 볼 수 있다.

표 3. 문맥 교환 시간 간격
Table. 3. Context switching time interval.

Q	40 MHz	100 MHz
100000	2.5 ms	1 ms
10000	250 μs	100 μs
1000	25 μs	10 μs



(a) eqntott

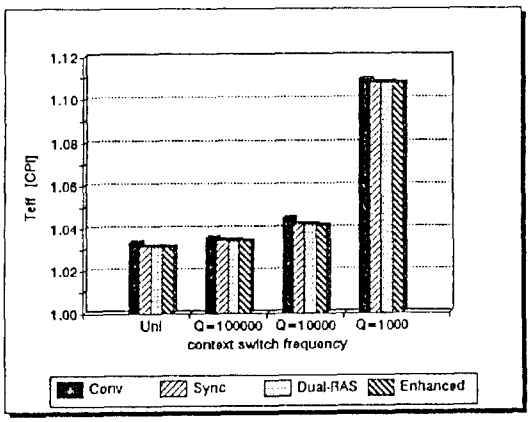


(b) spice3

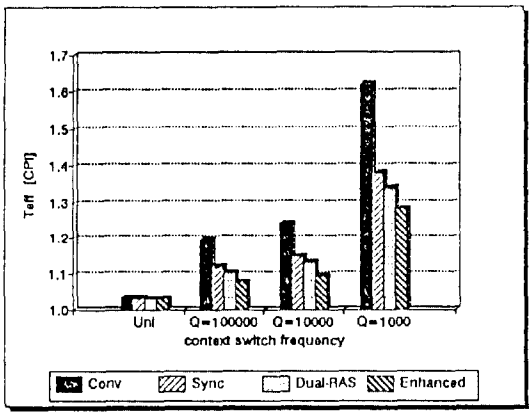
그림 7. 저가격 시스템에서의 멀티프로그래밍 효과
Fig. 7. Multiprogramming effect of a low cost system.

그림 7, 그림 8은 플러쉬 (flush)가 일어나는 명령어 간격 Q가 100000, 10000, 1000 인 경우에 대하여 각각 eqntott, spice3로 모의 실험을 수행한 결과이다. 저가격 시스템의 경우 단일 프로그래밍의 결과와 비교할 때 1-단계 캐쉬에서의 접근실패가 증가하므로 Teff가 증가하게 되지만 DRAM 간의 성능 향상 폭은 Q가 10000인 경우 1~3% 정도 더 추가된 성능 향상을 보여 준다. Q가 커질수록 새로운 구조의 DRAM들 간에도 성능 차이가 커져 enhanced DRAM이 synchronous DRAM에 비하여 3~4% 정도 높은 성능 향상을 나타냈다.

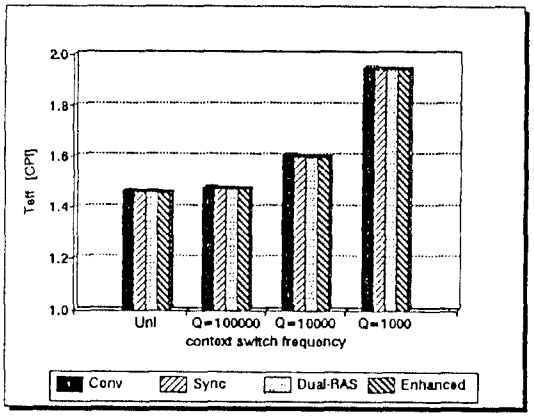
고성능 시스템의 경우에는 2-단계 캐쉬를 플러쉬시킨 경우와 플러쉬시키지 않은 경우에 대하여 큰 차이를 나타내는데 이것은 2-단계 캐쉬의 접근실패가 증가할수록 DRAM 접근의 빈도가 증가하게 되고 따라서 DRAM의 성능이 전체 시스템의 성능에 미치는



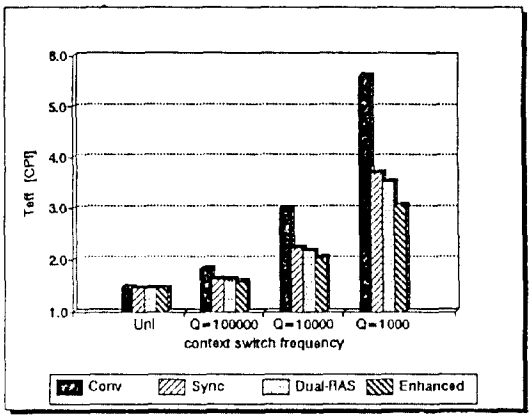
(a) 2-단계 캐쉬를 플러쉬시키지 않은 경우, eqntott



(b) 2-단계 캐쉬를 플러쉬시킨 경우, eqntott



(c) 2-단계 캐쉬를 플러쉬시키지 않은 경우, spice3



(d) 2-단계 캐쉬를 플러쉬시킨 경우, spice3

그림 8. 고성능 시스템에서의 멀티프로그래밍 효과
Fig. 8. Multiprogramming effect of a high performance system.

영향이 커진 결과이다.

실제로는 프로세스가 바뀔 때마다 2-단계 캐쉬 전체가 플러쉬되지는 않으므로 고성능 시스템의 결과의 해석에는 주의를 기울여야 한다. 만약 프로세스의 수가 적어 문맥 교환이 일어나더라도 2-단계 캐쉬의 데이타가 그대로 남게 된다면 (warm start의 효과) 그림 8. (a), (c)와 같이 거의 성능 차이가 없을 것이며 프로세스의 수가 증가하고 문맥 교환이 빈번해질수록 2-단계 캐쉬의 데이타를 잃어버릴 (replace) 확률이 높아지므로 그림 8. (b), (d)와 같이 20~45% 정도의 성능 차이를 보일 수 있다. 실제 평균적

으로 일어날 수 있는 상황은 Q=10000 정도이며 2-단계 캐쉬가 빈번히 플러쉬된다고 가정할 때 10~28% 정도의 성능 차이가 예상된다. 이 경우 새로운 DRAM들 간의 성능 차이는 5% 이내이다.

Ⅵ. 결론

새로운 DRAM 구조에 의한 메모리 시스템의 성능 개선은 멀티프로그래밍 효과를 고려하지 않으면 저가격 시스템에서 약 5~30% 정도로 기대되지만 고성능 시스템에서는 1% 이내로 예상된다. 그러나 실제의 경우 멀티프로그래밍의 효과를 고려하면 저가격 시스템의 경우 단일프로그래밍에서보다 1~3 % 정도 더 추가된 성능 향상이 기대되고 고성능 시스템의 경우에는 2-단계 캐쉬의 접근실패 증가로 인한 DRAM에의 접근이 빈번해짐에 따라 새로운 구조의 DRAM에 의한 성능 향상이 10~28% 내외이며 45% 정도까지 이를 수 있는 것으로 나타났다. 또한 새로운 DRAM 구조들 간의 성능 차이는 5%를 넘지 않는 것으로 나타났다. 따라서 기존의 DRAM으로 구성된 주메모리를 새로운 구조의 DRAM으로 교체하는데 소요되는 비용 증가가 전체 시스템의 성능 향상에 비하여 크지 않음을 가정하면 가격 대 성능비에서 경쟁력이 있음을 알 수 있다.

參 考 文 獻

[1] R. Comerford and G. F. Watson, "High Speed DRAMs," *IEEE SPECTRUM*, pp. 34-49, October, 1992.

[2] J. D. Gee, M. D. Hill, and A. J. Smith, "Cache Performance of the SPEC Benchmark Suite," *Technical Report UCB/CSD 91/648*, Computer Science Division, University of California, Berkeley, October, 1991.

[3] A. J. Smith, "Cache Evaluation and the Impact of Workload Choice," *Proceedings of the 12th Annual Symposium on Computer Architecture*, pp. 64-73, June, 1985.

[4] A. J. Smith, "Sequential Program Prefetching in Memory Hierarchies," *IEEE Computers*, pp. 7-21, December, 1978.

[5] D. A. Patterson and J. L. Hennessy, "Computer Architecture : A Quantitative Approach," *Morgan Kaufmann Publishers*, CA, 1990.

[6] A. J. Smith, "Cache Memories," *ACM Computing Surveys*, Vol.14, No.3, pp. 473-530, September, 1982.

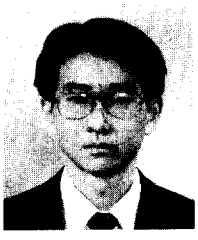
[7] W. H. Wang, J.-L. Baer, and H. M. Levy, "Organization and Performance of a Two-Level Virtual Real Cache Hierarchy," *Proceedings of the 16th Annual Symposium on Computer Architecture*, pp. 140-148, June, 1985.

[8] P. Y.-T. Hsu, "Introduction to SHADOW," Revision A of 28, Sun Microsystems, Inc. July, 1989.

[9] J. C. Mogul and A. Borg, "The Effect of Context Switches on Cache Performance," *Proceedings of the Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 75-84, 1991.

[10] A. Agarwal, J. L. Hennessy, and M. Horowitz, "Cache Performance of Operating System and Multiprogramming Workloads," *ACM Transactions on Computer Systems*, Vol. 6, No. 4, pp. 393-431, November, 1988.

著 者 紹 介



安 台 源(正會員)

1992年 서울대학교 전자공학과 학사. 1994年 서울대학교 전자공학과 석사. 1994年 ~ 현재 삼성전자 연구원

鄭 德 均(正會員)

1981年 서울대학교 전자공학과 학사. 1984年 서울대학교 전자공학과 석사. 1989年 U.C.Berkeley 전기 및 Computer 공학박사. 1989年 6月 ~ 1991年 8月 Texas Instruments VLSI Design Lab. 연구원. 1991年 8月 ~ 현재 서울대학교 공과대학 전자공학과 조교수

민 상 렬(正會員)

1983年 서울대학교 전자계산기공학과 졸업. 1985年 서울대학교 대학원 전자계산기공학과 석사. 1989年 워싱턴 주립대학 전산학 박사. 1989年 ~ 1990年 IBM T.J. Watson Research Center 객원연구원. 1990年 ~ 1992年 부산대학교 컴퓨터공학과 조교수. 1992年 ~ 현재 서울대학교 컴퓨터공학과 조교수. 주관심 분야는 Computer Architecture, Parallel Processing, Computer Performance Evaluation.

최 윤 호(正會員)

1983年 서울대학교 공과대학 전기공학과 졸업. 1983년 ~ 현재 삼성전자 반도체 연구소, 메모리 설계담당 과장