버퍼 시스템을 내장한 새로운 플래쉬 메모리 패키지 구조 및 성능 평가

(A New Flash Memory Package Structure with Intelligent Buffer System and Performance Evaluation)

이 정 훈 † 김 신 덕 ††
(Jung-Hoon Lee) (Shin-Dug Kim)

요 약 이 연구는 공간적/시간적 지역성의 효과를 이용하기 위하여 SRAM 버퍼를 사용하는 고성능 NAND-Type 플래쉬 메모리 패키지의 설계에 관한 것이다. 제안된 SRAM 버퍼를 내장한 새로운 NAND 형 플래쉬 메모리 패키지 구조는 크게 세 부분으로 구성되어 있다. 즉, 작은 블록 크기의 완전 연관 희생 버퍼(victim buffer)와 큰 블록 크기를 지원하는 완전 연관 공간 버퍼(spatial buffer), 그리고 동적 페칭 유닛(dynamic fetching unit)으로 구성되어 있다. 제안하는 새로운 NAND 형 플래쉬 메모리 패키지는 기존의 NAND형 플래쉬 메모리 구조와 비교할 때 매우 뛰어난 성능 향상 및 저 전력 소비를 이끌어낼 수 있다. 시뮬레이션 결과에 따르면 제안된 NAND 플래쉬 메모리 패키지는 기존의 NAND 플래쉬 메모리와 비교하여 접근 실패율에서는 70%, 평균 메모리 접근 시간에서는 67%의 감소 효과를 보여준다. 더욱이 주어진 크기(e.g., 3KB)의 SRAM 버퍼를 이용한 제안된 패키지는 여덟 배 크기의 직접 사상 버퍼(e.g., 32KB)를 이용한 패키지 및 두 배 크기의 완전 연관 버퍼(e.g., 8KB)를 이용한 패키지보다도 평균 접근 실패율 및 평균 메모리 접근 시간에서 더욱 우수한 성능 향상을 이끌어낼 수 있다.

키워드 : 플래쉬 메모리, 캐쉬 메모리, 시간적 지역성, 공간적 지역성, 시뮬레이션

Abstract This research is to design a high performance NAND-type flash memory package with a smart buffer cache that enhances the exploitation of spatial and temporal locality. The proposed buffer structure in a NAND flash memory package, called as a smart buffer cache, consists of three parts, i.e., a fully-associative victim buffer with a small block size, a fully-associative spatial buffer with a large block size, and a dynamic fetching unit. This new NAND-type flash memory package can achieve dramatically high performance and low power consumption comparing with any conventional NAND-type flash memory. Our results show that the NAND flash memory package with a smart buffer cache can reduce the miss ratio by around 70% and the average memory access time by around 67%, over the conventional NAND flash memory configuration. Also, the average miss ratio and average memory access time of the package module with smart buffer for a given buffer space (e.g., 3KB) can achieve better performance than package modules with a conventional direct-mapped buffer with eight times(e.g., 32KB) as much space and a fully-associative configuration with twice as much space(e.g., 8KB).

Key words : flash memory, cache memory, temporal locality, spatial locality, simulation

1. 서 론

오늘날 다양한 형태의 정보화/미디어 기기의 사용이 급속히 확대되고 있으며, 특히 이동 통신 단말기, MP3 재생기, 포터블 컴퓨터, 셋톱 박스, PDA와 같은 다양한 포터블 시스템의 출현과 보급은 기존의 DRAM 메모리 사용을 대신할 새로운 형태의 저장 장치에 대한 필요성의 증대로 이어지고 있다[1,2]. 디지털 카메라, MP3 플레이어 등 요즘 유행하는 최신의 디지털 미디어 기기들은 하나같이 편리함과 작은 사이즈를 자랑하는 최신 전자 제품으로, 필름이나 테이프에 의존하는 아날로그 방식에서 벗어나 디지털 데이타로 처리하는 최신 기술의 집약체이다. 이러한 디지털 방식의 각종 기기들은 환영

† 비 회 원 : 경상대학교 전기전자공학부 공학연구소 연구원
leejh@gsnu.ac.kr
†† 정 회 원 : 연세대학교 컴퓨터과학과 교수
sdkim@cs.yonsei.ac.kr
논문접수 : 2003년 6월 19일
심사완료 : 2004년 11월 23일

한 영상, 녹음한 음악 데이타가 디지털화 되어있기 때문에 데이타를 보관할 수 있는 별도의 미디어를 필요로 한다. 더군다나 디지털의 특성상 빠른 데이타의 입-출력을 필요로 하여 이제껏 많은 기술 개발과 다양한 미디어가 언급되기도 하였다. 디지털 데이타를 보관할 수 있는 미디어로는 광학미디어, 고밀도화된 마그네틱 타입의 미디어와 플래쉬 메모리의 미디어로 나누고 있다. 미디어는 각각의 장치 특성에 맞춰 여러 용도로 쓰이고 있는데, 현재 디지털 카메라, 캠코더, MP3 플레이어 등의 휴대용 장치에는 초소형의 기기들이기에 부피를 덜 차지하며 이동 중 데이타 보존을 위해 플래쉬 메모리가 가장 많이 쓰이고 있다. 플래쉬 메모리의 장점은 비휘발성이라는 특징과 고장 위험 감소 및 대용량 저장매체로써 그 중요성 및 수요의 대한 요구를 증대시키고 있다 [3-5].

주로 데이타용으로 사용되던 기존의 NAND 타입의 플래쉬 메모리는 NOR 타입의 플래쉬 메모리에 비해 가격이 저렴하다는 큰 특징을 가지고 있지만 읽기 연산 속도가 길어지는 치명적인 단점을 내포하고 있다. 오늘날 고성능 내장형 프로세서의 보급과 사용으로 플래쉬 메모리의 접근 시간에 대한 차는 현격히 증가되고 있는 추세이다. 또한 동영상 및 3D 게임, 화상 통신과 다양한 응용 프로그램들의 이동 통신 단말기에서의 사용 가능으로 사용자에게 고성능을 보장하기 위한 핵심 기술 중의 하나가 이러한 저장매체의 고성능이라 할 수 있다 [6]. 플래쉬 메모리의 이러한 단점 극복 방안으로 기존의 플래쉬 메모리에 쓰기 버퍼(write buffer)를 추가하여 쓰기 동작 및 지움(program/erase operation) 동작을 줄임으로써 그 성능을 높이는 연구가 활발히 진행중이다. 그러나 내장형 프로세서의 명령어 및 데이타 사용에 대한 내장 캐쉬 접근 실패(cache miss)의 경우 긴 접근 시간을 내포하는 플래쉬 메모리의 접근으로 이어져 전체 시스템의 성능을 급격히 감소시키고 최악의 경우(worst case) 프로그램의 수행을 원활히 동작시키지 못하도록 한다. 특히 화상이나 동영상과 같은 실시간 응용프로그램의 경우 플래쉬 메모리의 잦은 접근은 원활한 동작 수행을 어렵게 만드는 주요 원인이 된다.

이에 본 연구는 쓰기 동작 및 읽기 동작까지 모두 고속으로 처리 가능한 새로운 형태의 NAND 플래쉬 메모리 패키지를 제안하고자 한다. 즉, 제안하는 NAND 플래쉬 메모리 패키지는 기존의 NAND 플래쉬 메모리에 희생 버퍼와 공간적 버퍼를 가지고 있으며 동적 할당 메커니즘을 이용하여 지능적으로 블록 크기를 패치 하는 메커니즘을 사용하고 있다. 희생 버퍼는 쓰기 동작을 효과적으로 줄일 수 있으며, 공간적 버퍼는 읽기 동작을 고속으로 처리할 수 있는 역할을 수행하게 된다. 또한 동적 할당 메커니즘은 실시간적으로 프로그램의 특성을 파악하여 지능적으로 블록을 패치 함으로써 시간적 지역성 및 공간적 지역성을 효과적으로 이용하여 성능 향상의 효과를 얻을 수 있다.

시뮬레이션 결과에 따르면 제안된 구조의 SRAM 버퍼를 이용한 NAND형 플래쉬 메모리 패키지가 현재 상용 NAND형 플래쉬 메모리보다 접근 실패율을 약 70% 정도 감소시키고, 평균 메모리 접근 시간을 약 67%정도 감소시키는 것으로 나타났다. 또한 주어진 버퍼 크기(e.g., 3KB)의 SRAM 버퍼는 평균 메모리 접근 시간에서 여덟 배 이상의 직접 사상 버퍼(e.g., 32KB)와 두 배 이상의 완전 연관 버퍼(e.g., 8KB)보다 더 좋은 성능 향상을 이끌어 내는 것으로 나타났다.

이 논문의 구성은 다음과 같다. 2장에서는 플래쉬 메모리의 대표적인 형태인 NAND형과 NOR형의 특징과 차이점을 설명한다. 3장에서는 제안된 구조와 향상된 NAND 플래쉬 메모리의 성능에 관련된 기술에 대해 논한다. 4장에서는 시뮬레이션 결과로 나타난 성능에 대한 평가를 한다. 마지막으로 5장에서 결론을 맺는다.

2. 관련 연구

플래쉬 메모리는 크게 NAND형과 NOR형으로 구분된다. NAND형과 NOR형 모두 비트를 저장하기 위한 셀의 구조는 동일하지만 셀의 집합을 어떻게 구성하느냐에 따라 차이가 있고, 이로 인해 각각 성능 면에서 장/단점이 존재한다[7].

NAND형과 NOR형의 플래쉬 메모리의 특징을 종합하면 다음 표 1과 같다[7,8]. 일반적으로 상용화된 NOR

표 1 NAND형과 NOR형의 플래쉬 메모리의 특징

	NAND-type flash memory	NOR-type flash memory
장점	- 빠른 쓰기 동작 - 빠른 지우기 동작 - 작은 블록 사이즈 - 대용량화가 용이	- 빠른 random access - byte 단위로 random write가 가능
단점	- 느린 random access - byte 단위로 write가 불가능	- 느린 쓰기 동작 - 느린 지우기 동작 - NAND에 비해 대용량화가 어려움

플래쉬 메모리는 주로 실행을 위한 코드를 저장하는데 사용되고, NAND 플래쉬 메모리는 주로 데이타를 저장하는데 사용된다. 이는 NAND 플래쉬 메모리의 응용 가능성을 저하시키는 요소로 작용하게 된다. 또한 NOR 플래쉬 메모리는 읽기 동작에 있어 페이지/버스트(PAGE/BURST) 모드를 지원하는데, 페이지 읽기 모드는 시작 주소부터 정해진 크기의 데이타를 버퍼에 저장하는 방법으로 비동기적 읽기에 있어서 성능 향상을 목적으로 하며, 버스트 읽기 모드는 초기 접근된 데이타를 전송하는 작업과 동시에 다음 접근될 주소를 참조하여 접근하는 방법으로 동기적 읽기에 있어서 성능 향상을 목적으로 한다. 그리고 NOR 플래쉬 메모리의 적은 용량을 보완하기 위해 기존에 하나의 비트를 저장하는 셀에 두개의 비트를 저장하는 밀러비트(Mirrorbit)와 같은 기술을 이용하여 용량을 두 배 증가시키는 등, 성능/용량 측면에서의 개선이 이루어지고 있다. 그러나 NAND 플래쉬 메모리는 상대적인 단점인 읽기 성능을 개선시키는 연구는 이루어지지 않고 있는 실정이다. 그러나 NAND 플래쉬 메모리의 가장 큰 장점은 NOR 플래쉬 메모리보다 가격 면에서 약 40%정도 저렴하다. 심지어 NAND 플래쉬 메모리가 128KB 혹은 256KB의 SRAMs (static random access memory)를 집적화시켜도 그 가격이 상용 NOR 플래쉬 메모리와 거의 비슷하다. 게다가 NAND 플래쉬 메모리의 쓰기 접근 시간은 NOR 플래쉬 메모리보다 20배나 더욱 빠르다[7,9]. 이러한 이유로 고 밀도의 용량과 쓰기 명령 위주의 멀티미디어 기기들은 하드디스크나 NOR 플래쉬 메모리 대신 NAND 플래쉬 메모리를 사용한다. 하지만 NAND 플래쉬 메모리의 중요한 결점은 한 페이지 범위를 넘어서 읽기 명령을 수행하는 무작위 읽기 명령(random read operation)으로 그 수행속도는 NOR에 비해 100배 이상 차이가 나며, 실시간 응용 프로그램을 수행함에 있어서 원활한 동작을 하지 못하는 치명적인 단점으로 지적되고 있다. 그러므로 제안하는 새로운 형태의 NAND 플래쉬 메모리 패키지는 작은 비용을 추가하여 기존의 NAND 플래쉬 메모리 셀의 접근 및 쓰기 동작을 줄임으로써 성능 향상 및 소비전력에 효과적인 구조이다.

현재 상용 NAND 플래쉬 메모리의 프로그래밍(쓰기)과 읽기 명령은 페이지 단위 기반으로 수행된다. 이에 반해 지우기 명령은 블록 단위 기반으로 수행된다. 읽기 명령은 무작위(random) 방식과 연속적(serial) 페이지 읽기 방식의 두 가지 방식이 지원된다. 연속된 페이지 읽기 방식은 일반적으로 한 페이지 단위(e.g., 512-byte) 내의 연속적인 데이타 접근에 기반 한다. 이는 공간적 지역성을 효과적으로 이용할 수 있다. 하지만 대부분의 프로그램의 경우 메모리 참조 패턴의 경우 주어진 시간 내에 불러온 한 페이지의 모든 데이타를 이용하지는 않는다. 그러므로 이 방식은 버퍼 공간 및 전송 시간의 낭비를 초래하는 결과를 야기한다. 또한 무작위 읽기 명령은 페이지 주소가 바뀔 때에 실행되며, 최악의 경우 512Byte의 페이지 중 오직 하나의 워드나 단지 몇 개의 워드만 접근하고 새로운 페이지가 페이지 레지스터에 업데이트 될 수도 있는 것이다[10]. 그러므로 오직 하나의 페이지 레지스터를 지닌 상용 플래쉬 메모리 구조는 페이지 경계를 뛰어넘는 반복 접근 양태(threading effect)에서 시간적 지역성의 효과를 이끌어 낼 수가 없다. 이러한 결점을 극복하기 위해 다중 엔트리를 지닌 새로운 버퍼 캐쉬 시스템을 이용하여 시간적 지역성을 보다 효과적으로 이용할 수 있다. 또한 이 시스템은 페이지 블록 페치(page block fetch) 크기를 동적으로 조정하여 업데이트 시간을 줄이고 공간적 지역성의 효과를 높이고자 하였다.

최근 NAND 플래쉬 메모리를 주 역할인 데이타 저장 용도로써 사용하는 것이 아니라 주로 NOR 플래쉬에서 처리하는 명령어 부분까지 원만하게 수행하기 위하여 거대한 SRAM 또는 SDRAM(32KB~512KB)를 집적시켜 성능 향상을 얻고자 한 연구가 시도되었다[11]. 즉 NOR 플래쉬 메모리에서 담당하던 명령어 부분을 SRAM과 NAND 플래쉬 메모리만을 사용하여 실시간 응용 프로그램을 원활히 수행하기 위한 연구로써 NOR 플래쉬를 제거시킨 저비용의 플래쉬 메모리 패키지를 구현하였지만 소비전력이 높고 상대적으로 면적 및 비용이 높은 SRAM의 큰 용량을 사용함에 따라 이러한 플래쉬 메모리 패키지를 장착한 이동 단말기의 전체 소비전력에 치명적인 단점을 보일 수 있다. 그러나 제안하는 NAND 플래쉬 패키지 구조는 주 역할인 데이타 저장에 관련된 부분에 대한 연구로써 데이타 처리를 더욱 빠르게 수행할 수 있는 새로운 패키지 구조라 할 수 있다. 그러므로 쓰기 동작이 없는 명령어 부분은 기존의 NOR 플래쉬 메모리가 담당하고, 데이타 부분은 NAND 플래쉬 메모리가 담당하는 모바일 시스템의 한 형태로써 사용될 수 있다. 여기에 SRAM의 크기를 최소화하면서 새로운 인출 메커니즘 및 구동 메커니즘을 이용하여 적은 크기로 고성능을 보장할 수 있는 방식을 사용함으로써 기존의 NAND 플래쉬 메모리 자체를 고성능화하고 코어의 변화나 DRAM의 사용 유무와 관계없이 그 자체적으로 플래쉬 메모리 접근을 막아줌으로써 소비전력 및 전체 시스템의 성능을 올리고자 하는데 그 목적이 있다.

3. 새로운 NAND형 플래쉬 메모리 패키지 구성과 작동 모델

이 장에서는 연구 개발 동기에 대하여 설명하고, 아울러 SRAM 버퍼를 내장한 새로운 NAND 플래쉬 메모리에 대한 구조적 모델 및 블록 동적 할당 메커니즘의 구체적인 동작에 대해서 설명한다.

3.1 새로운 NAND 플래쉬 패키지 제안 동기 및 방법

이 연구에서 우리의 주 목표는 구조적으로 단순하고 저 전력 소비 효과 및 고성능을 보이는 NAND형 플래쉬 메모리 시스템을 개발하여 빠른 읽기 쓰기 동작을 지원하고 시간적, 공간적 지역성의 이용을 향상시키는 데에 있다. 상용 NAND 플래쉬 메모리는 하나의 페이지 레지스터만을 사용하여 오직 한 페이지 내의 공간적 지역성만을 이용하고 있다. 그러나 시간 간격을 두고 나타나는 공간적 지역성에 대해서는 전혀 그 효용을 이끌어 낼 수가 없다. 이러한 제약이 새로운 버퍼 시스템을 플래쉬 메모리에 패키지화하여 극복될 수 있다. 이 새로운 버퍼 시스템은 다양한 블록 크기를 동적으로 할당하는 기법을 통해서 다양한 응용 프로그램에 따라 서로 다른 시간적, 공간적 지역성에 알맞게 적응을 하게 된다. NAND 플래쉬 메모리의 새로운 구성은 뒤에 설명하는 원리로 이러한 목표를 이루게 한다.

전체 시스템의 저 전력 소비와 빠른 접근 시간을 보장하기 위해서는 플래쉬 메모리의 접근 자체를 줄여야 한다. 특히, 많은 무작위 읽기 명령과 쓰기 명령은 높은 전력 소비와 긴 실행 시간을 초래한다. 만일 모바일 기기가 영화나 음성 서비스 같은 다양한 실시간 응용 프로그램을 실행시킨다면, 이러한 요인에 의해 일정한 재생 시간이 보장되지 못하고 또한 기기의 배터리 수명을 감소시킬 것이다. 그러므로 플래쉬 메모리 접근 대신에 빠른 수행 시간을 보장하는 SRAM 버퍼가 먼저 접근되도록 하여 전체 프로그램의 빠른 수행 시간 및 저 전력 소비를 보장할 수 있다.

시간적, 공간적 지역성을 지닌 데이타를 위해 우리는 다양한 블록 페칭 크기를 지원하는 공간적 버퍼를 사용한다. 블록의 페칭 크기는 128B, 256B, 384B, 그리고 512B를 동적으로 할당하며, 이러한 선택적 크기는 실시간 동안 동적 페칭 유닛을 통하여 이루어진다. 공간적 버퍼 시스템을 설계하는데 있어서의 일반적인 목적은 바로 응용 프로그램의 속성에서 기인한 시간적, 공간적 지역성의 효과를 이끌어내는 것이다. 그러나 단일 블록 크기를 고수하는 것은 시간적 지역성과 공간적 지역성의 서로 양립하는 특성 때문에 이 둘의 효과를 최적으로 이용할 수가 없다. 블록 크기의 증가는 주어진 캐쉬 공간에서 캐쉬 블록 수의 감소로 귀결된다. 그러므로 상용 캐쉬 설계는 적절한 블록 크기를 택함으로서 타협한다. 그러나 상이한 타입의 지역성을 띄는 메모리 접근에 대하여 캐쉬의 적응력이 부족하면, 이는 더 심각한 결점으로 확장될 수 있다. 그러므로 고정된 블록 크기는 전체 접근 시간과 대역폭을 비 최적화 시키는 결과를 초래하며 프로그램 내에서 다양한 공간적 및 시간적 지역성을 이용하는데 적합하지 못하다고 볼 수 있다[12,13]. 또한 내장 캐쉬 메모리로부터 충돌 접근 실패(conflict miss)와 플래쉬 메모리의 쓰기 명령을 가급적 줄이기 위해, CPU내의 내장 캐쉬상에서 교체된 블록을 저장하기 위한 희생(victim) 버퍼를 사용한다[14]. 이 희생 버퍼의 블록 크기를 가능한 작게 만드는 것은 시간적 지역성을 향상시킴과 동시에 주어진 버퍼 크기 내에서의 활용 가능한 엔트리의 수를 증가시킴으로써 충돌 접근 실패와 쓰기 명령을 감소시킨다.

3.2 구조적 특징과 동작 모델

NAND 플래쉬 메모리와 SRAM 버퍼를 통합하고 구동시키는 새로운 알고리즘과 구조에 대해서 설명한다. 새로운 NAND 플래쉬 메모리 패키지의 구조는 그림 1과 같다. 이 패키지는 크게 NAND 플래쉬 메모리와 스마트(smart) 버퍼로 구성되며, 스마트 버퍼 시스템은 희생버퍼, 공간적 버퍼, 그리고 동적 페칭 유닛으로 구성되어 진다. NAND 플래쉬 메모리는 일반 상용 NAND 플래쉬 메모리 구조와 동일하게 구성되었다. 즉, NAND 플래쉬 메모리 셀들과 하나의 512-byte의 읽기/쓰기 레지스터로 이뤄져 있다. 스마트 버퍼 캐쉬는 내장 L1 캐쉬의 블록 크기와 동일한 32-byte 희생 버퍼(victim buffer)와 128-byte 블록을 지원하는 공간적 버퍼(spatial buffer)로 구성되어 있다. 공간적 버퍼의 블록 크기는 내장 L1 캐쉬 블록의 배수로 주어진다. 본 연구에서는 4배로 가정하였다. 상용 NAND 플래쉬 메모리

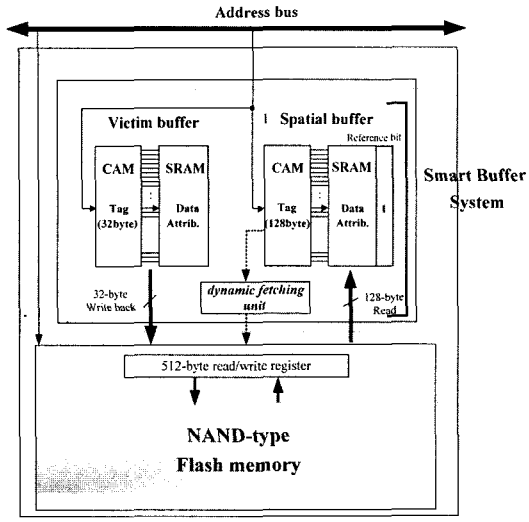


그림 1 스마트 버퍼 시스템을 내장한 새로운 NAND형 플래쉬 메모리 패키지 구조

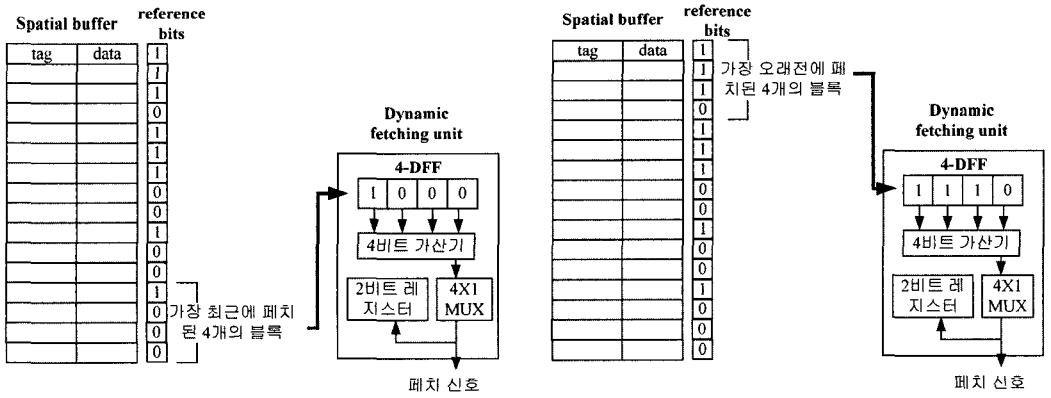
는 8핀의 단일 I/O포트로 지정되는 주소 범위를 갖는다. 이 방법은 핀의 개수를 크게 줄이고 시스템 보드의 호환성을 유지함으로써 훗날 업그레이드를 가능하게 한다. 따라서 제안하는 새로운 플래쉬 메모리 패키지 역시 이 호환성에 맞추어 설계되었다.

그러나 플래쉬 메모리와 스마트 버퍼 시스템 사이의 내부 버스의 폭은 32-byte의 대역폭으로 확장하였다. 희생 버퍼와 공간적 버퍼의 구조는 전형적인 완전 연관 구조와 동일하다. 두 버퍼는 모두 태그 저장을 위한 CAM(content addressable memory) 셀들의 집합과 데이타 저장을 위한 SRAM 셀의 집합으로 이뤄져 있다. 공간적 버퍼의 한 엔트리는 네 개의 연속적인 L1 캐쉬 블록의 집합으로 구성된다. 그러나 공간적 버퍼의 각각의 엔트리는 참조된 블록과 참조되지 않은 블록을 구분하기 위한 참조 비트(reference bit)를 포함한다. 이 참조 비트는 블록 페치 수행을 동적으로 생성하기 위해 사용된다.

CPU가 메모리 접근을 수행할 때 만일 CPU내의 온칩 캐쉬 상에서 접근 실패가 일어난다면, 희생 버퍼와 공간적 버퍼가 동시에 검색된다. 희생 버퍼나 공간적 버퍼 상에서의 적중(hit)은 일반 L2 캐쉬에서의 적중과 같은 방식으로 처리된다. 즉, 요청된 데이타를 CPU로 보내는 동시에 L1 캐쉬에도 올려놓는 것이다. 요청된 블록이 CPU의 내장 캐쉬에 업데이트 될 때, 만일 내장 캐쉬로부터 교체되는 어떤 블록이 존재한다면, 그 블록은 희생 버퍼로 이동하게 된다. 계속해서 만일 해당하는 희생 버퍼로부터 교체되는 엔트리가 수정된(dirty) 블록이라면, 방출되는 블록의 내용이 512-byte 읽기/쓰기 레지스터에 저장된다. 희생 버퍼는 온칩 L1 캐쉬의 충돌 접근 실패와 플래쉬 메모리로의 쓰기 명령을 효과적으로 줄일 수 있다. 만일 희생 버퍼의 수정된 블록이 적중하는 경우가 생기면, 그 블록을 다시 한 번 온칩 L1 캐쉬로 이동하게 되고 따라서 플래쉬 메모리로의 쓰기 명령은 피할 수 있게 된다. 공간적 버퍼에서의 적중인 경우에는 단순히 버퍼 내의 해당 블록이 페치되어 내장 L1 캐쉬로 보내지게 되고 동시에 그 블록은 참조가 일어난 블록임을 나타내기 위하여 그 엔트리의 참조 비트가 1로 설정된다.

희생 버퍼와 공간적 버퍼 모두에서 접근 실패가 발생하는 경우 128-bytes(한 개의 엔트리), 256-bytes, 384-bytes 및 512-bytes의 네 가지 중 하나가 결정되어 그 크기만큼 512-bytes의 읽기/쓰기 레지스터에서 공간적 버퍼로 이동한다. 이러한 페치 크기 결정은 동적 페칭 유닛의 예측 결과에 따른다. 전술하였듯이 요청된 블록이 내장 L1 캐쉬로 이동될 때 교체 방출되는 내장 L1 캐쉬 블록이 존재하면 이 교체되는 블록은 희생 버퍼로 이동된다.

동적 페칭 유닛은 그림 2처럼 4 개의 D Flip-Flops (FFs), 2-비트 레지스터, 그리고 4-비트 가산기(adder)와 MUX로 구성되어진다. 두 가지 페칭 메커니즘이 존재한다. 첫 번째(a), 두 버퍼에서 접근 실패가 발생하면 네 개의 FFs는 공간적 버퍼내의 가장 최근 페치된 4개의 엔트리에 대한 참조 비트를 저장하게 된다. 그러나 최근 페치된 4개의 엔트리에 대한 참조 비트는 현재 페이지의 메모리 접근 경향만을 반영할 뿐임으로 어느 정도의 시간 간격을 두고 발생하는 그 블록의 적중 정보는 반영하지 못한다. 그러므로 두 번째(b) 알고리즘은 네 개의 FFs으로 하여금 공간적 버퍼에서 교체될 수 있는 4개의 후보 블록들에 대한 참조 비트의 정보를 저장하도록 하였다. 즉 FIFO 대체 알고리즘에 의해 대치



(a) 가장 최근에 페치된 4개의 블록을 이용하여 페치 신호 생성

(b) 가장 오래전에 페치된 4개의 블록을 이용하여 페치 신호 생성

그림 2 동적 페칭 유닛의 구성도와 동작 알고리즘

될 블록들은 충분한 시간동안 버퍼 내에 존재하였으며, 프로그램의 수행동안 참조가 일어난 블록인지 아닌지를 명확히 판단할 수 있다. 이에 해당하는 엔트리들은 공간적 캐쉬의 가장 오래된 4개의 블록에 해당한다. 시뮬레이션 결과에서도 두 번째 알고리즘이 첫번째 알고리즘보다 더 좋은 성능을 지님을 알 수 있었다. 따라서 이 논문에서는 두 번째 알고리즘을 채택한다. 가산기는 네 개의 FFs에 저장된 모든 참조 비트들을 더해서 페치 신호를 도출해내기 위해 사용된다. 즉 두 버퍼의 접근 실패의 경우, 가산기의 결과는 블록 패치의 크기를 결정한다.

예를 들어 가산기의 결과가 0 혹은 1이라면, 페치 블록의 크기는 128-bytes이고 공간적 버퍼는 512-bytes의 읽기/쓰기 레지스터로부터 단지 128-bytes의 블록만을 가져올 것이다. 가산기의 결과가 3 혹은 4라면 페치의 크기는 384-bytes와 512-bytes(한 페이지 크기)가 된다. 예로 가산기의 결과가 1이라면 4개의 엔트리 중 단지 하나의 엔트리에서만 참조가 일어난 패턴임으로 이런 패턴 환경에서 512-byte 블록을 모두 페치하는 것은 무의미하다. 그러나 전체 프로그램이 모두 이러한 패턴을 보이지는 않을 것이다. 일반적으로 이런 패턴을 처음 보이기 시작하면 일정 시간동안 계속적으로 이러한 패턴을 보임을 알 수 있었다. 그러나 한 프로그램 내의 특정 부분에 대해서는 4개의 엔트리 중 2개를 참조하는 형태를 어느 시간 주기 동안 일정하게 나타날 수도 있고 또한 작업 셋(working set)인 큰 경우에는 4개의 엔트리 중 4개의 엔트리를 모두 참조하는 패턴을 보일 수도 있을 것이다. 그러므로 실시간으로 버퍼 접근 실패 시 매번 4개의 엔트리를 이용하여 판단함으로써 전반적인 프로그램의 작업 셋을 대략적으로 판단하여 레지스트로부터 블록을 페치 할 수 있는 것이다. 이러한 메커니즘에 더하여 2-비트 레지스터를 이용하여 좀 더 효율적인 페칭 메커니즘을 사용하고자 한다. 2-비트 레지스터는 이전 작업에서 패치 했던 블록 크기를 저장한다. 즉 이전 블록 페치 크기가 128-byte 인 경우 2-비트 레지스터는 00비트가 저장된다. 만약 256-byte, 384-bytes, 또는 512-bytes를 페치한 경우 2-비트 레지스터는 01, 10, 또는 11비트가 저장되어진다. 이 레지스터의 목적은 공간적 버퍼에서 참조되지 않은 블록들을 가능한 빨리 제거하기 위함이다. 즉 2-비트 레지스터에 00 비트가 저장된 경우, 이는 이미 설명한 것처럼 앞선 참조 실패 시 4개의 대치 블록 중 단지 하나의 참조 비트만 1로 셋팅된 경우이며, 4개중 3개의 블록은 충분한 시간 동안 한번도 참조가 일어나지 않은 블록이다. 이러한 블록들은 가능한 빨리 버퍼로부터 제거 시켜 주고 새로운 블록을 패치하여 버퍼에 저장시켜 주는 것이 더 효율적일 것이다. 이러한 동작을 수행하기 위하여 이전의 페치 크기가 128-bytes이고 현재의 패치 신호가 128-byte를 생성한다면, 실제 페치되는 크기는 128-byte가 아니라 256-byte로 확대된다. 마찬가지로 이전의 페치 크기가 256-bytes이고 현재의 패치 신호가 256-byte를 생성한다면, 실제 페치되는 크기는 256-byte가 아니라 384-byte로 확대된다. 그러므로 128-bytes 또는 256-bytes의 경우에 이러한 2-비트 레지스터의 사용은 보다 높은 성능 향상을 볼 수 있었다.

마지막으로 두 버퍼에서 접근 실패가 발생하면, 접근 실패한 워드를 포함하는 블록은 하위 레벨의 플래쉬 메모리로부터 공간적 버퍼로 올라가게 된다. 동적 페칭 유닛내의 참조 비트들과 2-비트 레지스터에 의하여 페칭 유닛은 다양한 크기의 블록을 페치한다. 그러나 접근 실패를 일으킨 블록(128byte) 외에 추가 페치되는 블록들이 공간적 버퍼 내에 이미 존재할 수도 있다. 그러므로 스마트 버퍼 제어기는 이러한 추가 페치 블록들이 존재하는지를 알아보기 위하여 공간적 버퍼의 태그를 재 탐색해야 한다. 만일 이러한 블록이 공간적 버퍼 내에 존재한다면, 해당하는 추가 페치 블록은 공간적 버퍼 내로 업로드 되지 않는다. 또한 희생 버퍼에 쓰기 동작이 이루어진 블록이 아직 쓰기 동작이 완료되지 않은 상태의 블록으로 공간적 버퍼에 존재할 수 있다. 이러한 일관성을 해결하기 위하여 희생 버퍼로부터 플래쉬 메모리로 쓰기 연산을 수행할 때 공간적 버퍼에 존재하는지를 알아보기 위하여 공간적 버퍼의 태그를 재 탐색해야 한다. 만약 존재하게 되면 쓰기 동작은 공간적 버퍼에서 수행되고, 존재하지 않을 경우에는 플래쉬 메모리로 쓰기 동작이 이루어진다.

4. 시뮬레이션과 분석적 모델을 통한 성능 평가

이 장에서는 시뮬레이션 환경 및 성능 평가에 대해 자세히 설명한다. 시뮬레이션은 트레이스 구동 방식을 택하였고, 사용된 벤치마크는 이미지 처리, 비디오 압축 및 해제, 음성 처리, 자연어 인식 등 실시간 응용 프로그램으로 대표되는 미디어 벤치마크를 사용하였다[15]. 이 벤치마크의 트레이스를 생성하기 위해 QPT2 시뮬레이터를[16] 사용하여 처리하였다. NAND 플래쉬의 주 저장 매체인 데이타 참조만을 시뮬레이션에 반영시켰다. DineroIV 캐쉬 시뮬레이터는[17] 제안된 버퍼 캐쉬 시스템을 시뮬레이션 하기 위해 수정되었다. 시뮬레이션 실행을 위한 시스템으로는 32KB 2-way 집합 연관 캐쉬를 지닌 200MHz MIPS를 사용하였다. 기본 시뮬레이션 환경 변수는 표 2와 같다.

4.1 페치 블록 크기의 효과

다양한 페치 크기를 초기화하기 위한 방법으로써 공

표 2 시뮬레이션 변수들

System parameters	Values
CPU clock	200 MHz
Random read time	10us
Serial read time	50ns
Buffer access time	85ns
Program time	300us
I/O port	8 bit
Flash memory internal bandwidth	32 byte

간적 버퍼내의 참조 비트들간의 최적 조합을 알아내기 위해서 시뮬레이션을 실행하였다. 간단하고 효과적인 방법은 설명한 전례에서처럼 단지 4개의 참조 비트들을 사용하여 얻을 수 있었다. 이러한 참조 비트 집합과 2-비트 레지스터의 정보에 의거하여 전술했던 바와 같이 4가지의 블록 페치 크기 중 하나가 결정된다. 이 메커니즘은 공간적 버퍼가 특정 응용 프로그램이 지니는 지역성의 패턴에 따라 동적으로 적응하게끔 한다. 그림 3은 실험에서 실제 초기화된 다양한 페치 크기에 따른 효과를 보여준다. Cjpeg을 제외한 미디어 벤치마크에서는 128-byte 페치 크기가 다른 경우보다도 우세함을 보여준다. 그러나 단일 128-byte나 512-byte 페치 크기가 사용될 때에는 더 큰 공간적/시간적 지역성을 보다 효과적으로 이용할 수가 없다.

상용 플래쉬 메모리에서 512-byte 블록 페치 기반은 이 블록내의 많은 데이타들의 접근 없이 대치되어짐으로 그 효용성이 낮은 것으로 드러났다. 하지만 상용 플래쉬 메모리에서 기본 페치 크기가 128-byte라면 이것은 512-byte 블록 기반보다 성능이 더 떨어짐을 보여준다. 이는 실시간 미디어 응용 프로그램은 시간적 지역성보다 공간적 지역성에 더 강한 경향을 보이기 때문이다. 그림 4는 이러한 경향을 잘 보여준다. 버퍼 공간의 효과를 나타내기 위해서 스마트 버퍼 시스템 대신에 상용 버퍼(캐쉬) 구조가 실험에 사용되었다. 이른바 직접 사상 버퍼(캐쉬)와 완전 연관 버퍼(캐쉬)의 두 가지 구조가 다양한 블록 크기에 대해서 실험에 사용되었다. 512-byte와 128 byte의 블록 크기를 지닌 플래쉬 메모리 패키지의 접근 실패율은 그림 4에서 보는 바와 같다. 여기서 나타난 접근 실패율은 내장 캐쉬를 포함한 시스템 전체의 접근 실패율이 아니라 플래쉬 메모리에 결합된 두 가지 구조의 버퍼의 접근 실패율이다.

직접 사상 버퍼는 DM으로 표기하였고 DM-512byte라는 표기는 페치 블록 크기가 512-byte인 직접 사상 버퍼를 뜻한다. 비슷하게 완전 연관 버퍼는 FA로 표기하였으며 FA-128byte는 128-byte의 블록 크기를 가진 완전 연관 버퍼를 의미한다. 여기서는 Epic 벤치마크의 결과만을 보여줬는데, 다른 벤치마크의 결과 역시 동일

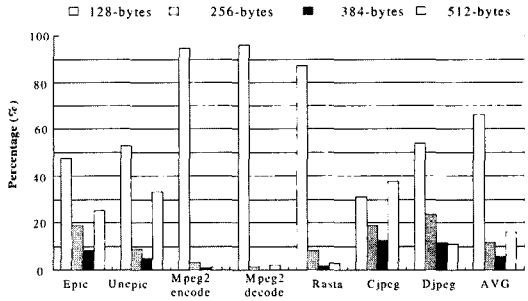


그림 3 다양한 블록 크기 페치 동작의 백분율

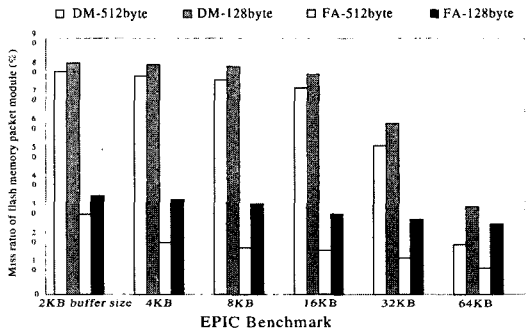


그림 4 다양한 버퍼 사이즈와 페치 블록 크기의 접근 실패율

한 버퍼 크기와 동일한 블록 크기에서 Epic 벤치마크와 비슷한 결과를 보였다. 결과에 따르면 직접 사상 버퍼구조를 플래쉬 메모리에 패키지화한 경우 32KB 버퍼 용량을 사용하고도 거의 성능 효과가 없음을 알 수 있다. 그러나 완전 연관 버퍼를 사용할 경우 128-byte의 경우 버퍼 크기에 관계없이 성능이 거의 일정함을 알 수 있으며, 512-byte 블록 페치의 경우 적은 버퍼 크기에도 불구하고 우수한 성능을 보임을 알 수 있다. 그러나 이러한 결과는 단지 접근 실패율에 대한 결과이다. 뒤에 설명하겠지만 평균 메모리 접근 시간의 경우 순수한 완전 연관 버퍼 구조가 성능 향상에 거의 영향을 미치지 못함을 알 수 있을 것이다.

결론적으로 직접 사상 구조와 완전 연관 구조 모두에서 512-byte 블록 기반이 128-byte 블록 기반보다 더 좋은 성능을 보여주고 있다. 또한 직접 사상 버퍼의 경우에는 버퍼 크기가 64KB 이상으로 구성될 때 의미 있는 성능 효과를 얻을 수 있다. 128-byte 블록 크기의 완전 연관 버퍼의 경우에서는 버퍼의 크기가 더 커지더라도 추가적이 성능 향상을 거의 얻을 수 없다. 그러나 512-byte 블록 크기의 완전 연관 버퍼는 버퍼 크기가 작을 지라도 접근 실패율의 관점에서 높은 성능을 보여준다.

그림 5는 제안한 모델과 메커니즘에 의한 성능 향상

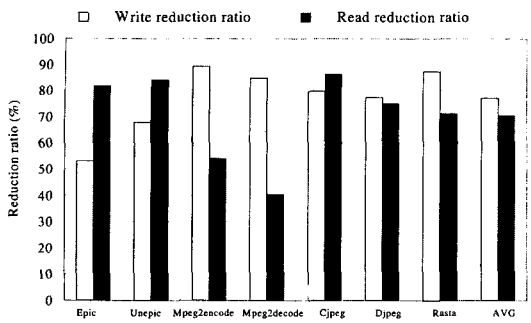


그림 5 기존의 NAND 플래쉬 메모리와 제안된 플래쉬 메모리 패키지의 비교 결과에 대한 효용성

정도를 기존의 버퍼 시스템이 없는 상용 NAND 플래쉬 메모리 모델에 대한 성능 향상 정도를 보여준다. 그림 5에서 알 수 있듯이 순차적 접근과 무작위 접근에 대한 읽기 명령과 쓰기 명령의 감소 비율을 보여준다. 이러한 결과는 제안된 패키지가 단지 3KB의 작은 버퍼만을 추가하고도 상용 플래쉬 메모리에 비해 쓰기 명령을 대략 78%, 읽기 명령을 대략 70% 정도 감소시키는 성능 향상을 얻을 수 있었다. 즉 전체적으로 70% 이상의 플래쉬 메모리 접근을 막을 수 있으며 이러한 성능 향상은 전체 시스템의 소비 전력 뿐만 아니라 응용 프로그램의 원활한 수행을 보장할 수 있는 성능 향상에 크게 기여할 수 있는 데이타로 인식되어진다.

4.2 상용 버퍼를 사용한 경우와 제안된 버퍼를 사용한 플래쉬 메모리의 성능 비교

메모리 시스템의 대표적인 성능 평가 지표인 접근 실패율(miss ratio)과 평균 메모리 접근 시간(average memory access time)이 제안된 스마트 버퍼와 다른 구조 방식들을 비교, 측정하기 위해서 사용되었다. 접근 실패율은 플래쉬 메모리 패키지 안의 버퍼 내에서 일치하는 블록이 없는 메모리 접근의 비율을 의미한다. 평균 메모리 접근 시간은 접근의 시작으로부터 요청한 데이타가 내장 캐쉬에 도착할 때까지의 지연된 시간을 의미한다. 플래쉬 메모리 패키지에서의 버퍼 크기는 접근 실패율을 결정하는 중요한 요소이다. 그러므로 이러한 버퍼 크기는 실제 설계 시 중요한 변수로써 작용되어진다. 버퍼 크기가 커질수록 시간에 영향을 미치는 주소 및 데이타 공간을 가로지르는 신호가 그만큼 길어지므로 버퍼는 느려진다. 또한 버퍼가 커질수록 전력 소비도 그만큼 더 늘어난다. 또한 비용도 무시하지 못하는 변수이다. 시뮬레이션에서는 32-bytes의 블록 크기를 갖는 1KB 희생 버퍼와 128-bytes의 블록 크기를 갖는 2KB 공간적 버퍼를 지닌, 즉 총 3KB의 버퍼 공간을 지니는 스마트 버퍼 시스템을 가정하였다. 32KB 또는 64KB의 크기인 직접 사상 버퍼와 4KB 또는 8KB의 크기인 완전 연관 버퍼가 제안한 스마트 버퍼 시스템과 비교 측정되었다. 그림 4에서 보인 바와 같이 고성능을 보장하기 위하여 상용 버퍼의 기본 블록 페치 크기는 512-byte로 가정하였다.

4.2.1 접근 실패율과 평균 메모리 접근 시간

제안된 스마트 버퍼 시스템에서 희생 버퍼와 공간적 버퍼의 최적의 블록 크기를 결정하기 위해 여러 가지의 실험을 수행하였다. 결과 32-byte의 작은 블록과 128-byte의 큰 블록이 대부분의 경우에 가장 좋은 성능을 보임을 시뮬레이션을 통하여 알 수 있었다. 플래쉬 메모리 패키지에 포함된 버퍼 시스템의 접근 실패율은 그림 6에 나타나 있다. 전술하였듯이 접근 실패율은 아래와 같은 식 (1)로 계산한다.

접근실패율=(버퍼내의 총 접근 실패수 / 전체 캐쉬 내에서의 접근 실패 수) *100 (1)

총 3KB 크기의 버퍼를 지닌 스마트 버퍼 시스템의 평균 접근 실패율이 대략 30%임을 알 수 있다. 이는 어떠한 버퍼 시스템도 내장하지 않은 상용 플래쉬 메모리와 비교하였을 때, 제안된 구조가 플래쉬 메모리 접근을 약 70%까지 줄일 수 있다는 얘기이다. 그리고 제안된 구조 및 메커니즘의 효율성을 검증하기 위한 시뮬레이션을 수행하였다. 기존에 이러한 플래쉬 패키지 구조에 대한 연구가 거의 전무한 상태이기 때문에 시뮬레이션에는 스마트 버퍼 대신에 기존의 직접 사상 버퍼와 완전 연관 버퍼를 플래쉬 메모리에 패키지화하였다고 가정하여 성능 평가를 수행 하였다. 이 세 종류 버퍼의 접근 실패율은 그림 6에 나타나 있다. 그림에서 알 수 있듯이 주어진 크기의(예로 3KB) 스마트 버퍼의 접근 실패율은 8배 크기의 직접 사상 버퍼보다도 좋은 성능 향상을 보이고 있으며, 2배 크기의 완전 연관 버퍼와 거의 비슷한 성능을 보여준다. 그러나 스마트 버퍼에서는 주 페치 크기가 128-byte 블록을 채택하는데 반해, 다른 구조에서는 512-byte 블록을 채택한다. 이는 로딩 시간(load time)이 그만큼 길어진다는 단점을 초래하게 된다. 메모리 계층 구조에서 성능을 평가하는데 주로 사용하는 또 다른 성능 평가 지표는 평균 메모리 접근 시간이다. 이는 다음과 같은 식에 의해 계산한다.

평균 메모리 접근 시간 = 캐쉬 적중 시간 + 버퍼 적중 시간 * 버퍼 적중률 + **접근 실패율 * 접근 실패 지연 시간** (2)

접근 실패율 * 접근 실패 지연 시간 = 순차적 읽기 비율 * 순차적 읽기 시간 + 무작위 읽기 비율 * 무작위 읽기 시간 + 쓰기 비율 *프로그래밍시간 (3)

여기서 적중 시간이란 캐쉬나 버퍼에서 적중했을 때의 수행 시간을 뜻하며, *접근 실패 지연 시간*이란 접근

실패가 발생했을 때 메모리 접근을 위해 지연되는 시간을 뜻한다. 시뮬레이션의 기본 변수들은 표 2에 수록하였다. 버퍼의 적중 시간은 상용화된 NAND형 플래쉬 메모리의 수행 시간에 따라 약 17cycle로 가정하였으며, 순차적 읽기 시간과 무작위 읽기 시간은 각각 10과 2000cycle로 가정하였다. 마지막으로 프로그래밍(쓰기) 시간은 60,000cycle로 가정하였다.

각각 스마트 버퍼와 상용 버퍼를 사용한 NAND 플래쉬 메모리 패키지의 총 평균 메모리 접근 시간은 그림 7에 나타나 있다. 총 평균 메모리 접근 시간은 CPU의 내장 캐쉬와 외부 플래쉬 메모리로 인한 결과를 모두 반영한다. 세 가지의 접근 명령 중에서 가장 결정적인 영향을 미치는 명령은 다른 두 개와 비교하여 훨씬 느린 쓰기 명령이다.

그림 6에서 보듯이 완전 연관 버퍼 구조의 평균 접근 실패율이 직접 사상 버퍼 구조보다도 더 좋은 성능을 나타낸다. 그러나 그림 7에서는 오히려 직접 사상 버퍼가 완전 연관 버퍼보다 평균 메모리 접근 시간에서는 더 좋은 성능을 보임을 알 수 있다. 이는 오직 작은 엔트리만을 지닌 완전 연관 버퍼가 많은 읽기 명령에 대해서는 접근 실패율을 줄여줄 수 있지만 상대적으로 높은 수행시간을 요하는 쓰기 명령을 거의 줄여주지 못하기 때문이다. 즉 쓰기 동작이 수행된 블록이 버퍼에서 다시 내장 캐쉬로 올라가지 못하고 읽기 동작에 의한 접근 실패로 버퍼에서 빠져 나와 바로 플래쉬 메모리로 업데이트 동작이 일어나기 때문에 많은 접근 실패율 감소에도 불구하고 전체 시스템의 성능에는 거의 영향을 못 미치게 되며, 상용 플래쉬 메모리와도 거의 유사한 성능을 보임을 알 수 있다. 이러한 결과는 버퍼의 크기가 전체 시스템에 중요한 변수가 됨을 알 수 있다. 그러나 제안한 스마트 버퍼 구조와 새로운 페칭 알고리즘은 단지 작은 크기의 버퍼 공간으로도 높은 성능을 이끌어 낼 수 있다.

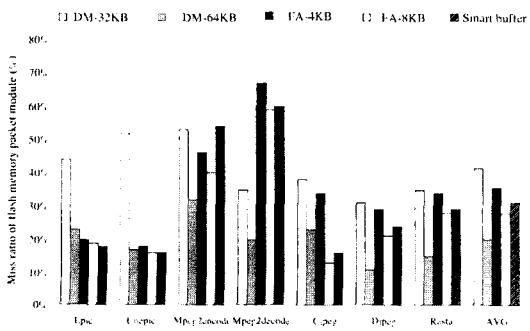


그림 6 기존의 버퍼 이용과 제안된 버퍼 사용에 대한 접근 실패율

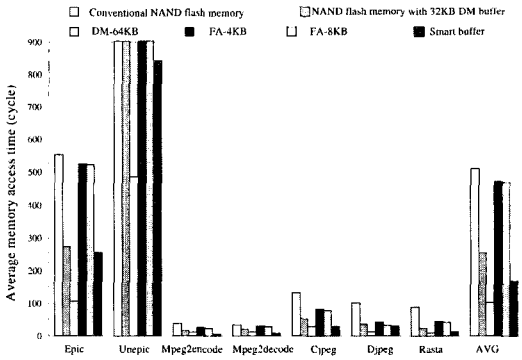


그림 7 기존의 버퍼를 이용한 NAND 플래쉬 메모리와 제안된 버퍼를 사용한 NADN 플래쉬 메모리에 대한 평균 메모리 접근 시간

5. 결 론

본 논문에서는 모바일 기기에 알맞은 저 비용의 간단한 하드웨어 기반의 높은 성능을 보이는 새로운 NAND형 플래쉬 메모리 패키지 시스템을 제안하였다. 제안된 NAND 플래쉬 메모리 시스템 내부의 스마트 버퍼는 완전 연관 희생 버퍼와 완전 연관 공간적 버퍼의 확장된 결합이다. 두 가지 지역성의 이점을 효과적으로 이용하면서 다양한 응용 프로그램에 적용 가능한 새로운 캐싱 기법이 소개되었다. 희생 버퍼는 충돌로 인한 접근 실패와 특히 많은 쓰기 명령을 효과적으로 감소시키는 역할을 한다. 공간적 버퍼는 공간적 지역성을 동적이고 지능적으로 이용할 수 있도록 고안되었다. 또한, 스마트 버퍼 시스템은 여러 가지 형태의 응용 프로그램에서 보이는 공간적 지역성의 수준에 따라 동적으로 블록 페칭 크기를 할당함으로써 성능을 향상시킨다. 즉, 접근 실패가 발생하였을 때 동적 페칭 유닛이 블록의 접근 경향을 기록한 정보에 따라, 네 가지의 페칭 크기 중 하나를 택하는 신호를 발생시키는 동적 메커니즘을 사용하였다.

시뮬레이션 결과에 따르면 제안된 새로운 NAND 플래쉬 패키지는 기존의 NAND 플래쉬 메모리 접근을 대략 70%정도 감소시키는 것으로 나타났다. 이 수치는 잠재적인 저 전력 소비와 고 성능의 수행을 보여주는 것임을 말해준다. 그리고 제안된 패키지에 단지 3KB의 버퍼 공간만을 추가함으로써 상용 플래쉬 메모리보다 쓰기 명령을 78%, 읽기 명령을 70% 감소시킬 수 있다.

마지막으로 스마트 버퍼를 이용한 NAND 플래쉬 메모리 패키지가 상용 NAND 플래쉬 메모리 보다 접근 실패율은 70%, 평균 메모리 접근 시간은 67%정도 감소시키는 것으로 나타났다. 또한 제안된 메커니즘을 사용할 경우 평균 메모리 접근 시간에서 여덟 배 이상의 기존 직접 사상 버퍼와(e.g., 32KB) 두 배 이상의 완전

연관 버퍼(e.g., 8KB)를 패키지화한 플래쉬 메모리 보다 더 좋은 성능 향상을 이끌어 내는 것으로 나타났다.

참 고 문 헌

[1] F. Douglis, R. Caceres, F. Kaashoek, K. Li, B. Marsh, and J. A. Tauber, "Storage Alternatives for Mobile Computers," In Proceedings of the 1st Symposium on Operating Systems Design and Implementation, 1994.

[2] N. Ballard, "State of PDAs and Other Pen-Based Systems," In Pen Computing Magazine, pp. 14-19, 1994.

[3] M. Baker, S. Asami, E. Deprit, J. Ousterhout, and M. Seltzer, "Non-Volatile Memory for Fast, Reliable File Systems," In Proceedings of the 5th International Conference on Architectural Support for Programming Languages and Operating Systems, 1992.

[4] B. Marsh, F. Douglis, and P. Krishnan, "Flash Memory File Caching for Mobile Computers," In Proceedings of the 27 Hawaii International Conference on System Sciences, 1994.

[5] B. Dipert and M. Levy, "Designing with Flash Memory," Annabooks, 1993.

[6] J. R. Lorch, and A. J. Smith, "Software Strategies for Portable Computer Energy Management," IEEE Personal Communications Magazine, 1998.

[7] Samsung Elec., "NAND-type Flash Memory," Http://www.samsungelectronics.com/semicon ductors/flash/Flash.html

[8] AMD Corp., "AMD Advanced Architecture Flash Memory Devices: Am29PDL128G(3.0v) Datasheet," http://www.amd.com/us-en/FlashMemory/ProductInformation/

[9] Toshiba America Electronic Component, "Flash memory," http://www.toshiba.com/taec/main /faq/flash_faq.html.

[10] M. L. Chiang, P. H. Lee, and R. C. Chang, "Flash Memory Management for Lightweight Storage Systems," Technical Report of Academia Sinica Institute of Information Science, TR-IIS-98-003, 1998.

[11] Chanik Park, Jaeyu Seo, Sunghwan Bae, Hyojun Kim, Shinhan Kim and Bumsoo Kim, "A low-cost memory architecture with NAND XIP for mobile embedded systems," In Proceedings of the 1st IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis, pp. 138-143, 2003.

[12] S. Przybylski, "The performance Impact of Block Sizes and Fetch Strategies," In proceedings of the 17th ISCA, pp. 160-169, 1990.

[13] F. Jesus Sanchez, Antonio Gonzalez, and Mateo Valeo, "Static Locality Analysis for Cache Management," In proceedings of the PACT97, pp. 261-271, 1997.

[14] N. P. Jouppi, "Improving Direct-Mapped Cache Performance by the Addition of a Small Fully Associative Cache and Prefetch Buffers," In proceedings of the 17th ISCA, pp. 364-373, 1990.

[15] C. Lee, M. Potkonjak, and W. H. Mangione-Smith, "MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communication Systems," MICRO-30, pp. 330-335, 1997.

[16] T. Ball and J. R. Larus, "Optimally profiling and tracing programs," In ACM Transactions on Programming Languages and Systems, Vol. 16, No. 4, pp. 1319-1360, 1994.

[17] J. Edler and M. D. Hill, "Dinero IV Trace-Driven Uniprocessor Cache Simulator," available from Univ. Wisconsin; ftp://ftp.nj.nec.com/pub/edler/d4/, 1997.

이 정 훈

1999년 성균관대학교 제어계측공학과(학사). 2001년 연세대학교 컴퓨터과학과(석사). 2004년 연세대학교 컴퓨터과학과(박사). 2004년~현재 경상대학교 전기전자공학부 공학연구원. 관심분야는 지능형 메모리 시스템, 저전력-고성능 시스템, 고성능 컴퓨터 구조임

김 신 덕

1982년 연세대학교 공과대학 전자공학과(학사). 1987년 University of Oklahoma 전기공학(석사). 1991년 Purdue University 전기공학(박사). 1993년 2월~1995년 2월 광운대학교 컴퓨터공학과 조교수 1995년 3월~현재 연세대학교 공과대학 컴퓨터과학과 교수(sdkim@cs.yonsei.ac.kr). 관심분야는 고성능 컴퓨터 시스템, 웹 컴퓨팅임