멀티미디어 내장형 시스템을 위한 저전력 데이터 캐쉬 설계

김 정 길[†] · 김 신 덕[††]

요 약

대용량의 데이터 처리가 요구되는 내장형 시스템에서 메모리의 비중은 아주 중요하며, 특히 제한적인 메모리를 최적으로 이용하기 위하여 응용의 특성을 활용하는 온칩(on-chip) 메모리 구조의 설계가 필요하다. 본 논문에서는 멀티미디어 응용을 위한 내장형 시스템에서 저전력을 위하여 작은 용량으로 설계되었으나 우수한 성능을 보이는 데이터 캐쉬(data cache)가 제안된다. 제안되는 캐쉬는 컴파일러의 도움 없이 구조적인 특징과 간단한 동작 메커니즘만을 이용하여 해당 응용의 데이터 지역성(data locality)을 효과적으로 반영할 수 있도록 작은 블록 크기를 지원하는 4KB 용량의 직접사상 캐쉬(direct-mapped cache)와 큰 블록을 지원하는 1KB 용량의 완전연관 버퍼(fully-associative buffer)로 구성되어 진다. 전체 5KB의 작은 캐쉬 용량으로 인한 성능 저하를 보완하기 위하여 멀티미디어 응용의 알고리즘 특성을 기반으로 응용 적응적인 다중 블록 선인출(adaptive multi-block prefetching) 기법과 효과적 블록 필터링(effective block filtering) 기법이 제안되었다. 시뮬레이션 결과에 따르면 제안된 5KB 캐쉬는 기존의 16KB 4-way 집합연관 캐쉬와 동등한 성능을 보이면서 소비 전력 면에서는 40% 이상의 감소를 보이고 있다.

키워드 : 데이터 캐쉬, 내장형 시스템, 멀티미디어, 저전력

An Area Efficient Low Power Data Cache for Multimedia Embedded Systems

Cheong-Ghil Kim[†] · Shin-Dug Kim[††]

ABSTRACT

One of the most effective ways to improve cache performance is to exploit both temporal and spatial locality given by any program executional characteristics. This paper proposes a data cache with small space for low power but high performance on multimedia applications. The basic architecture is a split-cache consisting of a direct-mapped cache with small block size and a fully-associative buffer with large block size. To overcome the disadvantage of small cache space, two mechanisms are enhanced by considering operational behaviors of multimedia applications: an adaptive multi-block prefetching to initiate various fetch sizes and an efficient block filtering to remove rarely reused data. The simulations on MediaBench show that the proposed 5KB-cache can provide equivalent performance and reduce energy consumption up to 40% as compared with 16KB 4-way set associative cache.

Key Words : Data Cache, Embedded System, Multimedia, Low Power

1. 서 론

최근 컴퓨팅 환경은 멀티미디어 응용의 확산에 맞추어 기존의 개인용 컴퓨터에서 PDA 또는 Smart-Phone과 같은 모바일 환경(mobile platform)으로 급속히 이동하고 있으며, 이러한 경향은 다양한 내장형 시스템(embedded systems)들이 폭넓게 시장에 확대되는 기회를 제공하였다. 일반적으로 멀티미디어 데이터의 처리는 많은 연산을 요구하는 알고리즘 기반으로 동작하게 되어 모바일 기기들에서도 고성능 내장형 시스템의 필요성은 더욱 증가하고 있다. 그러나 모바일 기기에서 고성능을 위한 단편적인 성능 개선의 노력은 전체적인 시스템 관점에서 성능의 저하를 가져올 수 있다. 이는 모바일 기기 설계 시 제한적 메모리 용량, 배터리 작동 시간, 전체 사이즈, 실시간 동작 등 제한적 요소들을 고려하여야 하기 때문이다.

내장형 시스템에서의 캐쉬 메모리는 범용 프로세서에서와 동일하게 고속의 프로세서와 저속의 주 메모리사이에서 메모리 접근 지연시간(memory access latency)과 소비 전력을 줄이고 전체 성능 향상을 위한 중요한 역할을 담당하고 있다. 특히 멀티미디어 데이터의 용량 및 복잡도가 증가하는 환경에서 응용 적응적 메모리 시스템의 설계는 매우 중요하며[1], 나아가 특정 응용에 특화되는 내장형 시스템에서 응용 적응적 온칩 메모리 시스템의 설계는 해당 응용에서의 예측 가능한 메모리 참조 패턴을 활용하여 최적의 결과를 얻을 수 있어 성과가 더 크다고 할 수 있다.

† 준 회 원 : 연세대학교 컴퓨터과학과 박사과정
†† 정 회 원 : 연세대학교 컴퓨터과학과 교수
논문접수 : 2005년 12월 12일, 심사완료 : 2006년 2월 22일

범용 프로세서의 캐쉬 성능은 캐쉬 용량을 증대시켜 캐쉬 적중률(cache hit ratio)을 높임으로 가능하다. 그러나 내장형 시스템에서 캐쉬 용량의 증대는 전체적으로 시스템 용량, 전력 소모, 또한 비용의 증가를 수반한다[2]. 특히 메모리 시스템에서의 전력 소모는 StrongARM 110에서 캐쉬 가 전체 전력 소모의 약 42%를 차지하는 연구[2] 결과를 고려하였을 때 저전력 캐쉬 설계의 중요성은 더욱 증대 된다. 결과적으로 동작 전력을 고려하여야 하는 내장형 시스템의 캐쉬 메모리는 가능하면 작은 용량으로 설계되어야 할 필요가 있다. 그러나 내장형 시스템에서 작은 용량의 캐쉬는 낮은 적중률을 야기하여 시스템 전체의 성능 저하를 가져올 뿐 아니라 전력 소모의 증대를 가져오는 문제점을 가져올 수 있다. 일반적으로 캐쉬 크기가 고정되었을 때 멀티미디어 응용에서 캐쉬 적중률을 증가시키기 위하여 예측 가능한 메모리 참조 패턴을 활용한 여러 가지 선인출(prefecth) 기법이 제안되고 있다[3, 4, 5]. 그러나 내장형 시스템에서와 같이 제한된 캐쉬 용량을 사용하여야 하는 경우 공격적 선인출 기법은 캐쉬 오염(cache pollution) 문제를 야기하여 전력 소모의 증대뿐만 아니라 전체적 성능의 저하를 가져오게 된다. 특히 캐쉬 오염 문제는 메모리 접근 중 규칙적 간격(unit stride) 형태의 접근이 높은 빈도를 나타내는 멀티미디어의 데이터 처리 시 심각성은 더해질 수 있다.

이러한 상반적인 메모리 시스템의 문제를 해결하고 멀티미디어 내장형 시스템에 특화시키기 위하여, 본 연구에서는 기본적으로 작은 용량의 캐쉬 설계를 목표로 멀티미디어 프로그램 수행 특성에 적합한 두 가지 지역성을 효과적으로 활용할 수 있는 저전력 데이터 캐쉬 구조를 제안하였다. 시간적 지역성(temporal locality)은 최근에 참조가 일어난 블록이 가까운 시간 내에 다시 참조될 확률이 높다는 것을 의미하며, 공간적 지역성(spatial locality)은 참조가 일어난 블록의 이웃블록이 참조될 확률이 높다는 것을 의미한다. 그러나 이러한 두 가지 지역성은 캐쉬 크기가 고정되어 있을 경우 서로 상반되는 특징을 가지고 있다. 즉 캐쉬 블록 크기가 커질수록 공간적 지역성은 효과적으로 반영할 수 있지만 캐쉬 엔트리(cache entry) 수의 감소로 시간적 지역성에 대한 효과는 감소된다. 그러나 제안된 캐쉬 시스템은 이러한 지역성을 효과적으로 반영할 수 있는 새로운 캐쉬 시스템이다. 그리고 작은 용량의 각 캐쉬 모듈들이 효과적으로 성능을 발휘할 수 있도록 멀티미디어 응용의 알고리즘 특성을 기반으로 응용 적응적 다중 블록 선인출(AMBP : adaptive multi-block prefetching) 기법과 효과적 블록 필터링(EBF : effective block filtering) 기법이 추가되었다. 특히 제안된 캐쉬는 접근 시간의 지연을 가져오지 않을 뿐 아니라 추가적인 하드웨어 자원을 많이 필요로 하지 않는다.

본 논문의 나머지 부분은 다음과 같다. 관련 연구는 2장에서 소개되어지며, 3장은 제안된 작은 용량의 캐쉬 구조와 멀티미디어 응용에 최적화시켜 성능을 향상시키기 위하여 고안된 기법들을 설명한다. 4장에서는 성능 평가 지표와 면적 대 성능 비교 그리고 소비 전력에 대한 실험 결과를 비교-분석한다. 마지막으로 5장에서 결론을 맺는다.

2. 관련 연구

멀티미디어 응용의 급속한 확산과 이에 따른 다양한 내장형 시스템이 소개되는 것에 비하여, 이와 관련한 상위 레벨에서의 캐쉬 구조에 대한 연구는 상대적으로 많지 않은 점이 있다. 특히 멀티미디어 데이터 처리 시 캐쉬의 효용성에 대한 초기 연구는 해당 데이터의 스트리밍 특징으로 작은 시간적 지역성, 비순차적 지역성, 그리고 필요한 전체 데이터가 1차 캐쉬에 로드 될 수 없는 대용량의 데이터 셋(data set)으로 구성되는 점을 들어 해당 응용군에서 데이터 캐쉬의 무용론까지 언급되었다[6]. 그러나 최근 연구에서는 멀티미디어 프로그램이 블록 분할 기반으로 동작한다는 특성에 기초하여, 기본적으로 구동 시 대용량의 데이터가 필요하지만 처리는 작은 블록 단위로 처리 가능하며, 이 작은 블록 안에 데이터의 재사용 가능성이 상당히 높아 지역성을 활용할 수 있어 캐쉬 동작에 유리하게 작동한다는 주장이 나오고 있다[7].

일반적으로 멀티미디어 응용에서 캐쉬 성능을 높이기 위한 연구는 데이터 참조 시에 나타나는 규칙적 간격의 메모리 참조 특성을 이용함으로 해당 데이터를 선인출하여 캐쉬 적중률을 높이는 방법 또는 별도의 공간을 활용하여 저장을 하는 방법이 많이 제시되어 왔다. 이러한 관련 연구로서 소프트웨어 엠펙 디코딩(software MPEG decoding)에서의 데이터 캐쉬의 성능 향상을 위하여 selective caching, cache locking, scratch memory, duplication reduction 등 하드웨어 기반의 기술이 제안되었다[8]. 이차원 선인출(two dimensional prefetch) 기법으로 neiboring prefetch 기법이 소개되었으며[5], 이는 멀티미디어 프로그램에서 일반적으로 채택되는 one-block look-ahead 기법 보다 우수한 성능을 보여주고 있다. 하드웨어 소프트웨어 복합 선인출(hybrid prefetch) 기법도 소개 되었는데[3], 이는 스트림 선인출(stream prefetch)을 위한 전용 명령어와 하드웨어 선인출 참조 테이블을 이용하여 자동으로 선인출을 수행 하도록 하였다. 하드웨어 전용 선인출 기법으로 스트림캐쉬(stream cache)와 스트라이드 예측 테이블(stride prediction table)을 병행하여 선인출하는 방법도 소개되었다[4]. 스트림캐쉬는 주 캐쉬에 추가된 작은 용량의 완전사상 캐쉬로서 선인출된 데이터가 저장되며, 캐쉬 적중 시 해당 블록이 주 캐쉬로 이동을 하지는 않는다.

이상 언급된 관련 연구에서 전반적으로 사용된 여러 형태의 선인출 기법은, 멀티미디어 데이터 참조 시 나타나는 규칙적 간격의 메모리 참조 동작을 이용함으로 캐쉬 적중률을 높이는 가장 효과적인 방법일 수 있다. 그러나 과도한 선인출 기법의 적용은 내장형 시스템에서 전체 시스템의 성능 저하를 가져오는 불합리성을 가져 올 수 있다. 이는 캐쉬 오염 문제뿐 아니라 각각의 참조와 병행한 주소 번역, 복잡한 컨트롤 로직, 그리고 추가적인 메모리의 필요성에 기인

한다. 이는 [9]에서 발견할 수 있듯이 기존의 내장형 시스템에서 주로 채택된 캐쉬 구조가 직접사상 캐쉬 또는 집합연관 캐쉬(set-associative cache)인 점에서도 알 수 있다.

본 연구에서 제안하는 캐쉬는 멀티미디어 내장형 시스템에 특화된 작은 용량의 저전력 데이터 캐쉬로서, 구조적 특징으로는 이중 캐쉬(dual cache)[10] 구조를 채택하였으며, 작은 용량으로 설계 시 나타날 수 있는 성능상의 문제점을 해결하기 위하여 간단한 하드웨어 기반의 선인출 기법과 블록 필터링 기법을 추가 하였으며, 그 결과 일반적으로 내장형 시스템에서 동급의 성능을 보여주는 비교 캐쉬보다 40% 이상의 전력 소모를 줄이는 성능 향상을 가져올 수 있었다.

3. 제안된 캐쉬의 구조적 특징과 동작 원리

이 장에서는 새로운 캐쉬 시스템을 제안하게 된 배경, 설계 방법 그리고 새로운 캐쉬의 동작 모델과 구조적 특징을 상세히 살펴 볼 것이다.

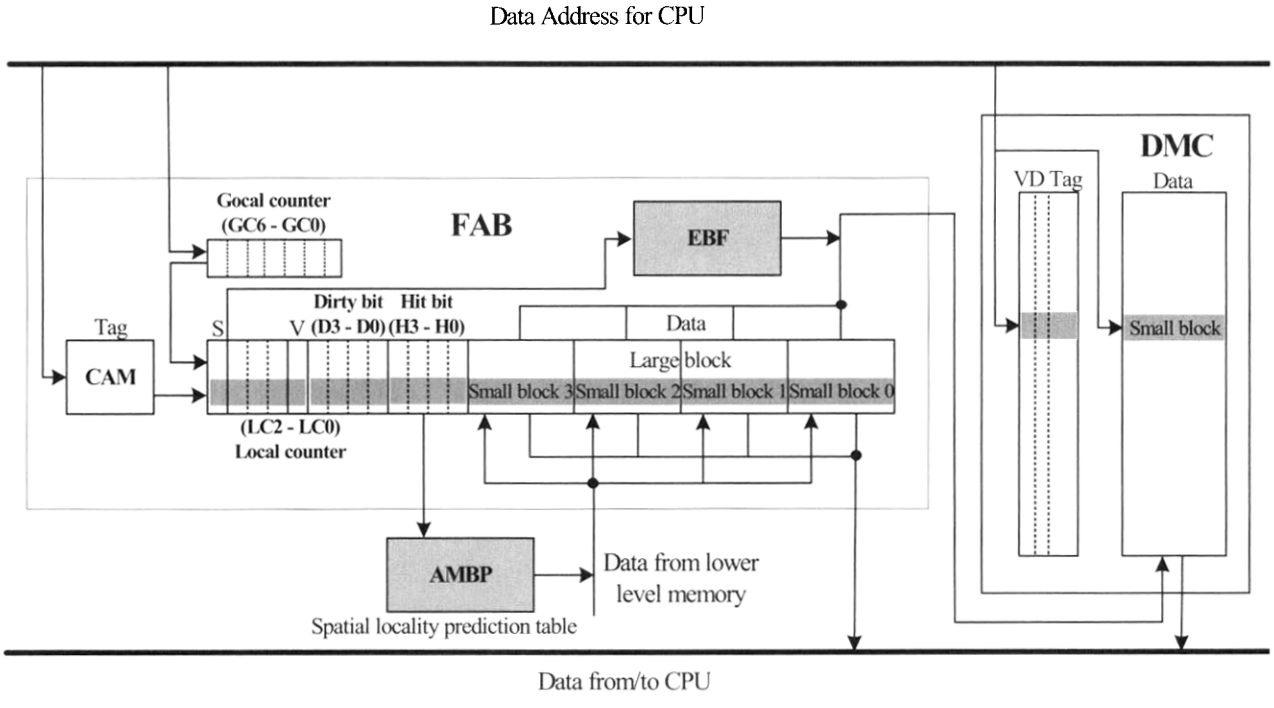
3.1 캐쉬 설계 동기 및 방법

제안된 캐쉬의 주목적은 멀티미디어 응용 프로그램에 적합한 시간적-공간적 지역성을 이용한 접근 실패 최소화 및 빠른 접근 시간 그리고 작은 용량의 저전력에 중점을 두고 설계 하였다. 기존의 하드웨어 기반의 선인출 기법을 채택한 캐쉬들이 이전 참조 어드레스의 저장을 통한 선인출 방법을 채택하였으나, 제안된 캐쉬는 지정된 스트라이드의 반복 횟수를 이용한 단순한 선인출 기법을 사용하였으며, 카운터를 이용하여 재사용 가능성이 높은 데이터를 필터링 함으로 캐쉬 충돌과 오염을 줄였다. (그림 1)은 제안된 캐쉬의 구조를 보여주고 있다. 기본적으로 상반된 특성을 가진 두 종류의 지역성을 효과적으로 이용하기 위하여 두 개의 분리된 캐쉬 구조를 선택하였으며 작은 용량의 각 캐쉬의 성능을 높여주는 간단한 부가적인 하드웨어가 추가되었다. 두개의 분리된 캐쉬 중에 작은 블록 크기를 가진 직접사상 캐쉬(DMC : direct-mapped cache)는 엔트리 수 증가에 의한 시간적 지역성을 반영할 수 있으며, 저전력과 빠른 접근 시간에 가장 적합한 구조로서 주 캐쉬로 선택하였다. 공간적 지역성을 반영하기 위하여 큰 블록을 가진 완전연관 버퍼(FAB : fully-associative buffer)를 같은 계층에 위치시켰다. 제안된 캐쉬 시스템이 두 모듈의 캐쉬로 구성이 되었다고 하더라도 내장형 시스템 설계 시 제한적 요소인 면적과 가격 등을 고려하여 전체 면적을 최소한 작게 설계하였다.

완전연관 버퍼(FAB)는 연속된 다수의 메모리 블록을 단일 인출 단위로 동작하도록 설계 되었다. 이는 캐쉬 블록 사이즈의 증가가 데이터 선인출의 효과를 가져와 공간적 지역성의 활용도를 높이는데 도움이 되며, 특히 높은 공간적 지역성을 내포하는 멀티미디어 데이터 경우 인출 블록 사이즈의 확대는 선인출 효과를 일반 응용 프로그램 데이터와 비교하여 더 얻을 수 있다. 제안된 캐쉬는 적중 실패 시 해당 블록 다음 블록을 추가적으로 인출하는 one block look-ahead 선인출 기법을 기반으로 멀티미디어 데이터의 높은 공간적 지역성을 고려하여 추가 불럭을 선인출 하도록 설계되었다. 그러나 모든 적중 실패 시 다중 블록을 선인출 하는 경우 버스 트래픽의 증가, 캐쉬 태그 및 데이터 포트의 추가적 동작, 캐쉬 오염도 증가 등의 단점들을[11] 수반할 수 있는 관계로, 캐쉬 적중 실패 시점의 공간적 지역성의 정도에 따라 동적으로 선인출 여부와 인출 블록 사이즈를 결정하는 적응적 다중 블록 선인출(AMBP : adaptive multi-block prefetching) 기법을 제안하였다.

적응적 다중 블록 선인출(AMBP)은 지역성 예측 테이블(SLPT : spatial locality prediction table)을 이용하여 실행된다. SLPT에는 3개의 지정된 스트라이드와 카운터 값이 저장되며, 각 데이터 참조 시 이전 어드레스와의 차이 값을 계산하여 SLPT에 저장되어 있는 스트라이드 값과 일치하는 경우 해당 스트라이드의 카운터 값을 증가하게 된다. 저장된 값들은 캐쉬 적중 실패 시점의 공간적 지역성 정도를 3개의 지정된 스트라이드가 2개의 연속된 적중 실패 사이에 반복된 횟수(inter-miss-stride-count)를 이용하여 수치화 하여 표현한다. 이 수치는 주 메모리로부터 데이터를 인출할 때 기본 블록에 추가하여 인출할 데이터의 사이즈를 결정하게 된다. 지정된 스트라이드의 반복 횟수가 많은 경우는 현재 프로그램이 공간적 지역성이 높은 모드에서 작동하는 것이고 반복 횟수가 적은 경우는 상대적으로 공간적 지역성이 낮은 모드에서 작동하고 있다고 정의하였다. 이 예측을 바탕으로 32, 64, 96 바이트의 인출 데이터 단위에서 하나를 선택하여 데이터를 인출한다. SLPT의 지정된 스트라이드 값은 첫째, 멀티미디어 프로그램이 8 또는 16 비트 정수 단위로 연산이 이루어지며, 둘째 데이터의 접근이 횡선 주사(raster-scan) 모드 또는 mask-based 모드 형태로 규칙적인 간격으로 이루어진다는 특징에 기반을 두어 결정하였다. 따라서 3개의 지정된 스트라이드는 8, 16, 32 비트 데이터의 참조를 고려하여 1, 2, 4로 지정하였다. 제안된 선인출 기법은 프로그램 및 컴파일러의 수정 없이 간단한 하드웨어 추가만으로 성취 가능하여, 검증 결과에서 볼 수 있듯이 전체 시스템에 추가적 overhead를 많이 발생 시키지 않는다.

직접사상 캐쉬(DMC)는 작은 블록 단위로 작동하는데, 완전연관 버퍼(FAB)에 저장되어 있는 큰 블록이 대치될 때 조건에 맞는 작은 블록을 선택적으로 저장시킴으로써 하나의 저장 공간(one set)만을 가진 DMC의 성능을 증가시키도록 설계되었다. 즉 FAB의 큰 블록은 DMC가 지원하는 작은 블록 크기와 동일한 블록들로 구성되며, FAB에서 대치되는 순간 과거에 참조된 이력을 바탕으로 다시 참조가 일어날 가능성이 높은 작은 블록만 필터링하여 DMC로 다시 저장하게 된다. 이때 사용되는 effective block filtering(EBF) 기법은 DMC에 저장될 블록이 FAB에 머무는 동안 최종 참조 시점이 임계값 내에 있는 블록만 선택적으로 저장함으로 직접사상 캐쉬의 시간적 지역성을 최대한 활용하도록 하였다. 이는 작은 용량의 DMC가 스레싱(threshing)에 의해 오염되는 문제를 방지하기 위함이며, 이때 제안된



(그림 1) 제안된 데이터 캐쉬 구조

EBF기법은 FAB의 각 블록에 참조 비트와 카운터를 두어 연속된 두 번의 메모리 참조 사이의 시간(inter-reference-elapsed time)을 측정하여 결정하였다. 결과적으로 FAB에서 블록 대치가 발생할 경우 해당 블록에 참조 비트가 설정되고 해당 카운터가 임계값보다 작은 경우에만 DMC에 재 저장하여 불필요한 데이터의 이동에 의한 작은 용량의 DMC가 오염되는 것을 방지하게 된다. 이러한 방식의 효과는 FAB에서 대치되는 블록 중 참조비트가 설정된 모든 블록을 DMC에 저장시키는 메커니즘에 비해 충돌에 의한 접근 실패와 스래싱 효과를 줄일 수 있고 시간적 지역성을 가진 데이터의 수명 시간을 연장시킬 수 있는 장점을 가지게 된다.

3.2 제안된 캐쉬의 구조

제안된 캐쉬의 구조는 (그림 1)과 같다. 직접사상 캐쉬(DMC)는 주 캐쉬의 역할을 담당하며 일반적인 직접사상 캐쉬와 동일한 구조로써 n-byte 블록 크기를 가지는 m개의 작은 블록으로 구성되어진다. 완전연관 버퍼(FAB)의 데이터 저장 공간은 여러 개의 뱅크(banks)로써 구성되어지며 각 뱅크의 블록 크기는 DMC의 블록 크기와 동일하다. 즉 선택된 상수 c로부터 $c \times n$ byte, k개의 큰 블록으로써 구성되어진다. 따라서 FAB의 총 바이트는 $k \times c \times n$ byte이다. 그리고 버퍼의 태그 부분은 모든 엔트리를 동시에 비교할 수 있는 CAM(content addressable memory)으로 구현되어지며 각 뱅크에 저장되어지는 블록들은 1비트를 가지고 있다. 이러한 비트는 큰 블록내의 작은 블록들 중에서 참조가 일어난 블록을 나타내기 위한 비트로 이후부터 적중 비트(hit bit)라 명명할 것이다.

완전연관 버퍼(FAB)의 각 블록은 태그 필드(tag field)와 카운터(counter) 필드를 필요로 한다. 카운터의 구조는 전역 카운터(global counter)와 지역 카운터(local counter)로 구성된 2단계 구조로 설계되었다. 여기서 카운터는 해당 블록이 참조 되고 다시 참조될 때까지의 발생한 메모리 참조 횟수를 표시하는데, 해당 값을 한 개의 카운터를 이용하여 저장할 경우 각 블록별로 많은 비트가 필요하게 된다. 따라서 전역 카운터는 매 백번의 메모리 참조 마다 자신을 리셋(reset)시키면서 지역 카운터가 증가되도록 신호를 발생시킨다. 각 블록에 설계된 지역 카운터는 3비트의 카운터 필드와 1비트의 세츄레이션 비트(saturation bit)로 구성되어 있어 최대 계산 가능한 참조 카운터는 $2^3 \times 2^{max\ number\ of\ bits\ of\ global\ counter}$가 된다. 세츄레이션 비트는 지역 카운터가 최댓값에 도달했을 때 지역 카운터가 리셋 되면서 세트 된다. 이러한 카운터 기반의 간격 측정 방법은 [12]의 실험 결과와 같이 프로세서의 속도에 큰 영향을 미치지 않는다. 지역 카운터는 3비트만 이용하는 관계로 태그 비교 또는 데이터 출력 시간과 병행하여 동작 가능하며 각 블록에서 차지하는 면적 비율은 1.5%에 불과하다. 전역 카운터는 백번의 카운트를 위하여 7비트가 필요하며 최대 비교 값을 저장하기 위하여 추가적인 래취(latch)가 필요하다. 다양한 시뮬레이션을 통하여 최댓값은 700이 최적이었으며, 이때 DMC와 FAB 사이에 약 40%의 데이터 이동을 감소시키는 효과를 가져왔다.

메모리 참조 실패 시 주 메모리로부터 인출할 블록의 크기를 결정하는 적응적 다중 블록 선인출(AMBP) 기법을 위해 설계된 공간 지역성 예측 테이블(SLPT)은 3개의 선 지정된 스트라이드 값에 해당하는 카운터 값을 저장하게 되는데, 각 메모리 참조 시 컨트롤러는 이전 메모리 참조 어드레스와 현재 참조 어드레스의 차이를 구하여 SLPT에 저장

되어 있는 스트라이드 값과 비교하여 동일한 경우 해당 카운터를 증가 시키게 된다. 그리고 캐쉬 적중 실패 시 SLPT의 카운터 정보를 이용하여 해당 프로그램의 공간적 지역성 정도를 판단하여 32, 64, 96 바이트 인출 단위 중 하나를 선택하여 실행한다. 실험을 통하여 세 개의 카운터 값 모두가 20 보다 큰 경우 인출 크기를 96 바이트로 선택하며, 반복 횟수가 1번 이상 19번 이하인 경우 64 바이트를 인출하며 그렇지 않으면 32 바이트 인출 크기를 유지하게 하여 제안된 캐쉬가 동적으로 해당 응용의 구동 특성에 적응적으로 선인출 작업을 수행하도록 하였다.

3.3 캐쉬 제어 알고리즘

여기서는 제안된 캐쉬 시스템의 동작 원리를 상세히 설명하고자 한다. CPU로부터 임의의 메모리 참조가 발생하게 되면 직접사상 캐쉬(DMC)와 완전연관 버퍼(FAB)가 같은 계층에서 동시에 접근이 일어난다. 이때 adaptive multi-block prefetching과 effective block filtering을 위하여 두 동작이 병행된다. 첫째 컨트롤러는 이전 참조 어드레스와 현재 참조 어드레스의 차에 의한 스트라이드를 구해 spatial locality prediction table을 갱신한다. 한편 각 블록의 지역 카운터의 동작을 위해 전역 카운터를 증강시킨다. 구체적인 가능한 동작의 경우는 다음과 같다:

㉠ **직접사상 캐쉬(DMC)에서의 적중:** DMC에서 적중이 일어나면 기존의 직접사상 캐쉬처럼 동일한 방법으로 CPU 요청을 수행하게 된다. 읽기 명령어인 경우 요청한 데이터를 CPU로 보내고 캐쉬의 동작은 끝나게 되며, 쓰기 명령어인 경우 해당 블록에 쓰기 동작을 수행함과 동시에 갱신 비트(dirty bit)를 1로 설정하게 된다.

㉡ **완전연관 버퍼(FAB)에서의 적중:** 메모리 주소가 CPU로부터 발생되어질 때 주소의 일부 비트를 이용하여 큰 블록 내의 여러 개의 뱅크들 중에서 부합되는 하나의 작은 블록만이 인출되어짐과 동시에 참조가 일어난 블록임을 나타내기 위하여 그 작은 블록의 적중 비트는 1로 설정되어 진다. 동시에 지역 카운터의 값들이 리셋 된다. 위에서도 언급했듯이 FAB의 데이터 부분은 뱅크로 나누어져 있기 때문에 디코더(decoder)를 이용하여 항상 FAB 참조 시 하나의 뱅크만 선택적으로 허가(enable)가 가능하다. 이것은 소비 전력을 줄이기 위한 방법이며 DMC와 FAB 모두 작은 블록 크기처럼 동작되어 질 수 있다. 만약 설계의 예처럼 작은 블록 크기가 8-byte이고 큰 블록 크기가 32-byte인 경우 뱅크의 수는 (그림 1)처럼 4개로써 구성되어진다. 따라서 한 번에 하나의 뱅크만 선택되어지am로 전력 소비적인 측면에서 유리한 장점을 가지게 된다. 만약 FAB내의 하나의 뱅크에서 적중이 발생되면 읽기 명령인 경우 적중이 발생한 뱅크의 작은 블록으로부터 요청한 데이터를 CPU로 보내고 동시에 그 블록의 적중 비트를 1로 설정한다. 쓰기 명령인 경우에는 참조가 일어난 블록에 쓰기 동작을 수행하고 갱신 비트와 적중 비트를 동시에 1로 설정한다.

㉢ **두 캐쉬에서 접근 실패:** 만약 DMC와 FAB에서 모두 접근 실패가 발생하면 캐쉬 제어기는 접근 실패 작업(miss handling)을 실행하게 된다. 기본적인 동작은 DMC에서 접근 실패가 발생한 작은 블록을 포함한 큰 블록이 다음 메모리 계층으로부터 인출되어 FAB에 저장되어진다. 예를 들어 작은 블록이 8-byte이고 큰 블록이 32-byte인 경우 32-byte 블록의 경계(boundary)에 속하는 4개의 순차적인 작은 블록이 인출되어 FAB에 저장되어지며, 이때 FAB의 유효 비트(valid bit)가 모두 1인 경우와 그렇지 않은 경우로 나누어진다.

- 완전연관 버퍼의 유효 비트가 모두 1이 아닌 경우 : 접근 실패가 발생하였을 때 FAB 내에 적어도 하나의 엔트리가 무효화 상태이면 다음 메모리 계층으로부터 큰 블록을 인출하여 버퍼의 비어있는 엔트리에 저장시킴과 동시에 참조가 발생한 작은 블록의 적중 비트와 유효 비트를 각각 1로 설정한다. 만약 읽기 접근 실패이면 요청한 데이터를 CPU로 보내고, 쓰기 접근 실패이면 쓰기 정정(write-back) 모드 방식으로 동작하게 되며 요청한 큰 블록을 FAB에 저장시키게 된다.
- 완전연관 버퍼의 유효 비트가 모두 1인 경우 : FAB의 대체 알고리즘으로 선입선출(FIFO) 방식을 채택하였다. 만약 FAB의 모든 엔트리가 유효 상태이면 접근 실패 시 가장 오래된 큰 블록이 선택되어지며 그 대체 블록에 속하는 작은 블록들 중에서 이미 참조가 일어난 블록을 선택하여 DMC로 저장하게 된다. 그러므로 FAB에서 직접 다음 계층 메모리로 쓰기 정정되는 경우는 발생할 수 없다. DMC 또한 이 경우 대치 블록이 발생하게 되며 그 대치 블록의 갱신 비트가 1인 경우 다음 계층 메모리로 쓰기 정정되며 갱신 비트가 0인 경우에는 무시하게 된다. 이러한 과정은 캐쉬 제어기가 접근 실패를 처리하는 동안 완료되어지기 때문에 추가적인 지연 시간은 발생되지 않는다. 그리고 큰 블록이 버퍼에 저장되는 방식은 위에서 설명한 경우와 동일하다. 기본 동작 외에도 제안된 캐쉬에서는 AMBP 기법에 의해 추가적인 인출 작업이 SLPT의 카운터 값을 이용한 예측 모드를 가동하여 64, 96 바이트를 인출할 수 있다. 이때 추가적인 블록이 이미 FAB에 존재하는지 검사가 실행된다.

4. 시뮬레이션과 분석적 모델을 통한 성능 평가

이 장에서는 시뮬레이션 수행을 통하여 제안된 캐쉬 구조에서 사용된 선인출 기법과 필터링 기법의 효율성과 접근 실패율과 평균 메모리 접근 시간을 다른 캐쉬와 비교 분석하고, 면적 대 성능 비 그리고 전력 소비에 대해서 분석적 모델을 사용하여 설명하고자 한다. 시뮬레이션 수행을 위하

여 SimpleScalar/ARM 프로세서 시뮬레이터[13]를 이용하여 3개의 벤치마크(MediaBench, MiBench, SPEC2000)의 트레이스 구동(trace-driven) 시뮬레이션을 수행하였다. 사용된 벤치마크는 각각 멀티미디어응용, 일반 내장형 시스템응용, 그리고 일반 시스템응용을 대표한다. 또한 SimpleScalar를 수정하여 L1 캐쉬의 메모리 접근 트레이스를 생성하였으며, 결과를 Dinero IV[14] 트레이스 구동 캐쉬 시뮬레이터를 이용하여 분석하였다. 비교 대상으로는 일반적인 기존의 직접사상 캐쉬와 내장형 프로세서의 주 캐쉬로 많이 사용되는 집합연관 캐쉬와 다양한 캐쉬 구조들 중 비교 분석 결과 가장 높은 성능 향상을 보였던 victim 캐쉬를 선택하였다.

4.1 블록 인출 사이즈의 구성 및 필터링 비율 분석

제안된 캐쉬 시스템의 완전연관 버퍼(FAB)는 큰 블록 인출과 추가적인 선인출 기법을 적용하여 캐쉬 적중 실패율(cache miss ratio)의 감소를 가져오는 반면에 메모리 접근 시간을 증대 시킬 수 있다. 이러한 문제점을 해결하기 위하여 제안된 adaptive multi-block prefetching 기법은 프로그램이 공간적 지역성이 높은 모드로 동작하는 경우 선인출시 추가적인 큰 블록을 인출 하고 그렇지 않은 경우는 선인출 신호를 발생하지 않게 함으로, 선인출 빈도를 줄이게 되며, 평균 메모리 접근 시간의 길어짐을 방지시켜 성능 향상을 가져온다. 그리고 직접사상 캐쉬(DMC)는 데이터를 선별하여 저장함으로 캐쉬 오염 문제를 방지하도록 effective block filtering 기법이 적용되었다.

<표 1>은 3개 벤치마크에서 각각 인출 요청된 블록 사이즈의 구성 비율을 보여 주고 있으며, 특히 멀티미디어 응용인 MediaBench에서 64 또는 96 바이트 인출의 빈도수가 훨씬 많은 것을 보여주고 있다. <표 2>는 블록 필터링 기법 적용으로 DMC로 재 저장될 블록의 제거 비율을 보여 주고 있다. MediaBench 경우 필터링 비율은 약 40%였으며 SPEC2000에서의 약 2%의 필터링 비율을 보여주고 있다. 이상의 실험에서 제안된 캐쉬 시스템에서 사용된 멀티미디어 응용에 특화된 두 가지 메커니즘의 성능을 가늠할 수 있는데, AMBP와 EBF 기법이 일반 응용 프로그램을 대표하는 SPEC2000에서 보다 멀티미디어 응용의 MediaBench에서 효율적으로 사용되었음을 보여주고 있다.

〈표 1〉 인출 요청 블록 사이즈별 구성

Benchmarks	Fetch signal distribution		
	32-bytes	64-bytes	96-bytes
MediaBench	41.72%	24.09%	34.17%
MiBench	43.95%	22.32%	33.79%
SPEC2000	79.40%	10.44%	10.15%

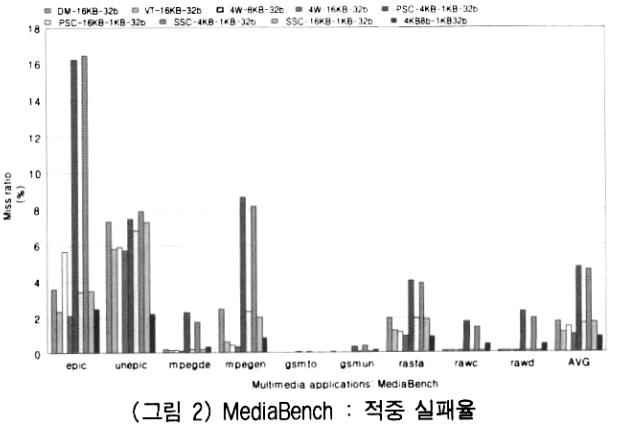
〈표 2〉 블록 필터링 비율

Benchmarks	Filtering ratio	
	To DMC	Trashed
MediaBench	55.29%	44.70%
MiBench	67.47%	32.53%
SPEC2000	98.23%	1.77%

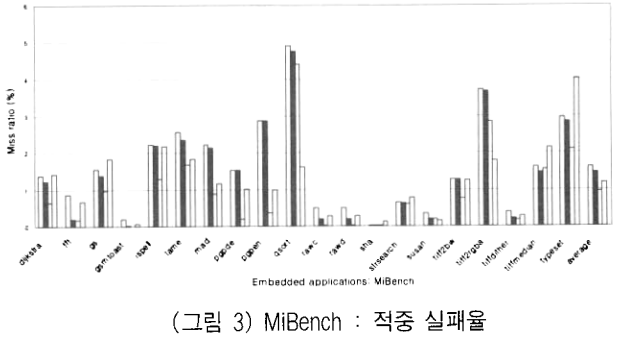
4.2 접근 실패율과 평균 메모리 접근 시간

일반적으로 캐쉬의 성능을 나타내는 지표로서 접근 실패율(miss ratio)과 평균 메모리 접근 시간(AMAT : average memory access time)을 사용한다. 본 실험에서는 내장형 시스템에서 일반적으로 사용되는 캐쉬 구조와 제안된 캐쉬와의 접근 실패율을 비교하게 된다. 기존의 직접사상 캐쉬는 DM로 표기하였으며 "16KB-32b"는 32-byte의 블록 크기를 가진 16KB 캐쉬 크기를 나타낸다. 2-way 집합연관 캐쉬와 4-way 집합연관 캐쉬는 각각 2W와 4W로 표시하였다. Victim 캐쉬는 VT로 표시되며 4개의 엔트리를 갖는 버퍼를 가정하였다. Parallel stream cache와 Serial stream cache는 각각 PSC와 SSC로 표시하였다. 이들 캐쉬의 블록 사이즈는 공통적으로 32-byte 이며 용량은 16KB이다. 그리고 제안된 캐쉬를 나타내는 "4KB8b-1KB32b"는 8-byte 블록크기를 가진 4KB 직접연관 캐쉬(DMC)와 32-byte 블록 크기를 가진 1KB 완전연관 버퍼(FAB)를 나타내고 있으며, 전체 용량은 5KB 이다. 실험은 각각의 벤치마크에 대해 이상 언급된 블록 크기와 캐쉬 크기로 수행하였다.

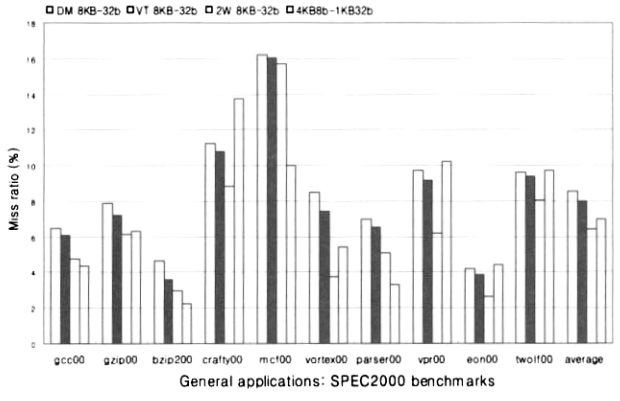
(그림 2, 3, 4)는 제안된 캐쉬와 다른 캐쉬들의 접근 실패율을 각 벤치마크 별로 표시하고 있다. (그림 2)는 MediaBench에서의 접근 실패율을 보여주는데 제안된 캐쉬가 전체 용량



(그림 2) MediaBench : 적중 실패율



(그림 3) MiBench : 적중 실패율



(그림 4) SPEC2000 : 적중 실패율

5KB로서 약 3배의 작은 용량으로도 16KB의 직접사상 캐쉬, 4-way 집합연관 캐쉬, victim 캐쉬와 우수한 성능을 보이고 있다. 제안된 캐쉬의 일반성을 검증하기 위하여, (그림 3, 4)에서 보여 주듯이 일반적 내장형 시스템 응용과 기존 일반 용용 시스템과의 비교를 위하여 MiBench와 SPEC2000에서도 접근 실패율을 실험을 실행하였다. 이때 비교 캐쉬들의 크기는 8KB 이며, 4-way 집합연관 캐쉬는 2-way 집합연관 캐쉬로 대체 되었다. 일반적인 용용에서도 제안된 캐쉬는 멀티미디어 응용에서 얻는 성능적 향상 보다는 다소 떨어지지만, 여전히 우수한 성능을 2-way 집합연관 캐쉬를 제외한 비교에서 보여주고 있다. 그러나 일반적으로 집합연관 캐쉬는 직접사상 캐쉬에 비해 접근 실패율을 크게 줄이는 장점을 가지는 구조이지만, 느린 접근 시간과 높은 전력 소비를 가져오는 단점을 가지고 있다.

실험 결과에서도 알 수 있듯이 제안된 캐쉬는 이상 다른 캐쉬들과 비교하여 멀티미디어 응용에서 우수한 성능을 보이고 있으며, 이후에 설명될 소비 전력까지 고려한다면 제안된 캐쉬 시스템이 더욱 효율적인 캐쉬 시스템임을 보여주고 있다.

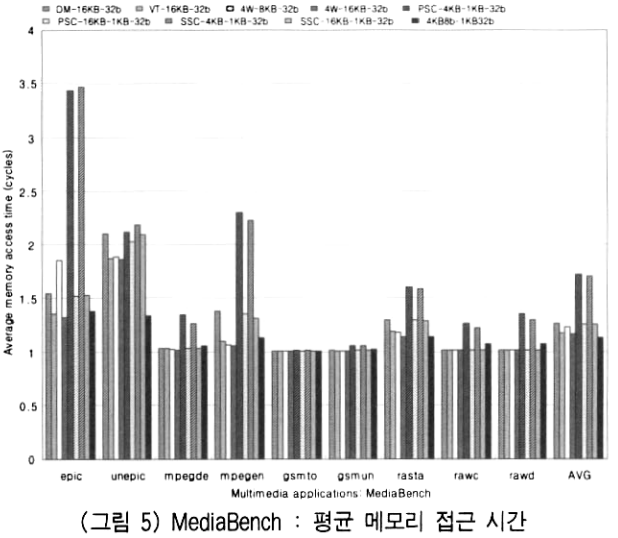
캐쉬 시스템의 성능 평가를 보다 정확하게 측정하기 위한 방법으로 평균 메모리 접근 시간(AMAT)을 이용하였다. AMAT를 얻기 위한 식은 다음과 같다.

$$\textit{Average memory access time} = \textit{Hit time} + \textit{Miss rate} * \textit{Miss penalty} \quad \text{(수식 1)}$$

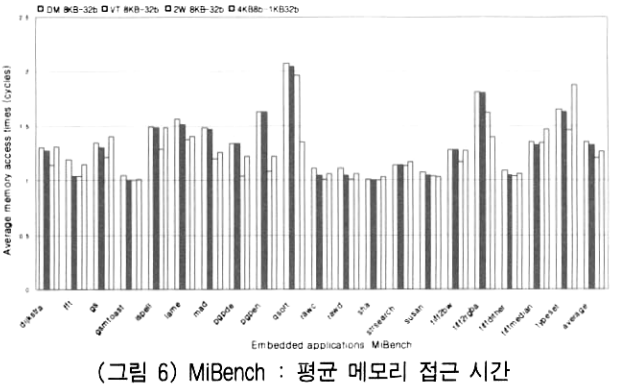
여기서 *hit time*은 캐쉬에서 적중을 처리하는데 걸리는 시간이며, *miss penalty*는 캐쉬 접근 실패 시 이를 처리하는데 추가되는 시간이다. 시뮬레이션을 수행하기 위한 구체적인 변수 값들은 <표 3>으로 정의되어진다. 이러한 변수 값들은 일반적인 32-bit 내장형 프로세서에서 사용되어지는 값들을 사용하였다(예로 Hitachi SH4 또는 ARM920T). 이를 기준으로 8-byte 블록이 64-bit 데이터 버스를 통하여 매 사이클 전송가능하며, 32-byte 블록을 인출하는 사이클은 22 클럭 사이클이 소요된다. 추가적으로 64-byte 또는 96-byte 블록을 인출하기 위하여 각각 34와 46 사이클이 소요되게 되며, 인출 전에 FAB에 해당 블록들이 이미 존재하는지 검사를 수행한다.

〈표 3〉 시뮬레이션 변수들

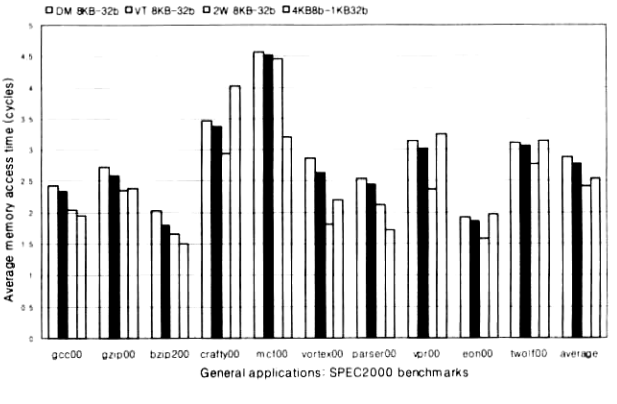
System parameters	Values
CPU clock	200 MHz
L2 cache	None
memory latency	15 /cpucycle
memory bandwidth	1.6 Gbytes / sec
DMC hit time	1 /cpucycle
FAB hit time	1 /cpucycle



(그림 5) MediaBench : 평균 메모리 접근 시간



(그림 6) MiBench : 평균 메모리 접근 시간



(그림 7) Spec2000 : 평균 메모리 접근 시간

4.3 면적 비용 대 성능 비에 대한 비교 및 전력 소비에 대한 분석적 모델과 비교 분석

이 장에서 제안된 캐쉬의 성능/비용과 전력 소비의 분석을 위하여 CACTI-III 시뮬레이터[15]를 이용하였으며, 프로세서 공정기술은 0.13μm, 전압은 1.3V로 가정하였다. <표 4>는 각 캐쉬들의 면적과 성능 비교를 보여주고 있다. 제안된 캐쉬는 비교 캐쉬들 중 가장 우수한 성능을 보여주고 있는 4-way 집합연관 캐쉬와 비교하여 약 48%의 작은 용량을 가지면서 동등한 성능을 얻고 있다. 표면적인 캐쉬 사이즈를 비교하였을 때 제안된 캐쉬가 약 3배의 감소를 가져옴에도 불구하고 실제적으로 약 2배의 향상을 가져오는 이유는 적응적 다중 블록 선인출과 블록 필터링 기능을 위하여 추가적인 컨트롤로직이 필요했기 때문이다.

추가적으로, 아래의 (수식 2)를 이용하여 캐쉬의 동적인 소모 전력을 측정하였다.

$$energy\_cache = (average\_power * average\_access\_time) * cache_{reference}count \quad \text{(수식 2)}$$

이때, *average_access_time* 과 *average_power*는 (수식 3)과 (수식 4)와 같이 각각 정의 된다.

$$average\_access\_time = \frac{\sum_{K=1}^{cache_{hit}count} access\_time_{access}(k) + \sum_{K=1}^{cache_{miss}count} access\_time_{miss\_handling}(k)}{cache_{reference}count} \quad \text{(수식 3)}$$

$$average\_power = \frac{\sum_{K=1}^{cache_{hit}count} power_{access}(k) + \sum_{K=1}^{cache_{miss}count} power_{miss\_handling}(k)}{cache_{reference}count} \quad \text{(수식 4)}$$

(수식 4)의 $power_{access}(k)$는 캐쉬 적중 시 캐쉬 블록에 접근하는데 소모되는 전력이다. 각각의 캐쉬들의 적중 방법에 따라 다른 $power_{access}(k)$ 값을 가지게 된다. 예를 들어 victim 캐쉬에서 캐쉬 적중이 victim 버퍼에서 발생했을 경우 스와핑 작업이 추가되므로 전력 소모가 증가하게 된다. 제안된 캐쉬에서도 캐쉬 적중은 DMC와 FAB에서 모두 발생하기 때문에 이에 따른 전력 소모는 틀리게 된다. <표 5>는 주요 비교 캐쉬에서의 접근 소모 전력을 보여 주고 있다.

캐쉬 전체의 전력 소모를 얻기 위하여 [9]에서 수행한 연구를 참조하여 (수식 5)를 이용하여 제안된 캐쉬에 분석적 방법으로 적용하였다.

$$overall\_energy = number\_of\_hits * hit\_energy + number\_of\_misses * miss\_handling\_energy \quad (5)$$

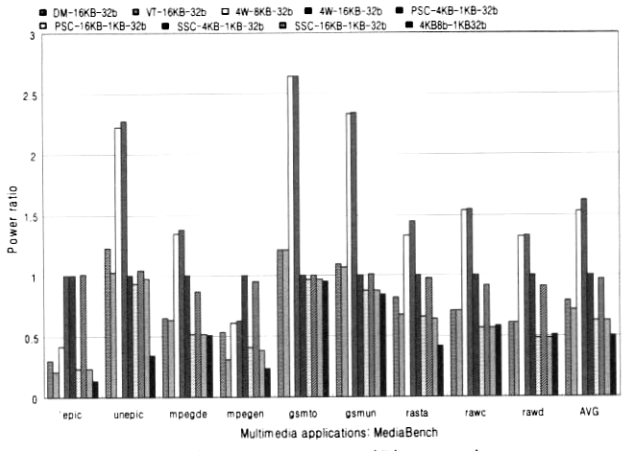
분석적 전력 소모 모델에 의하면 캐쉬 적중 실패 시 필요로 하는 소모 전력은 캐쉬 적중 때 보다 50배에서 200배 추가적으로 요구되는 것으로 정의되어 있다. (그림 8)은 각 비교 캐쉬들의 전체적인 소모 전력에 대한 비교를 보여주고 있다. 이때 사용된 전력 소모 모델은 최소 비율인 '50'으로서, '200'에 대한 비교 결과는 '50'의 결과에 일정비율로 전력 소모가 증가된 결과를 가져와 보다 큰 성능 격차를 보여주어 그림 작업을 생략하였다. 결과적으로 제안된 캐쉬는 비교 캐쉬들과 비교하여 40% 이상의 소모 전력의 감소를 가져오고 있다. 이러한 차이는 근본적으로 제안된 캐쉬가 작은 용량으로 설계되었으나 멀티미디어 응용의 특성에 기반을 둔 간단한 부가적 메커니즘을 전체적인 시스템의 오버헤드를 최소화 하면서 추가하여 얻은 결과이다. 이는 제안된 캐쉬가 저전력을 요구하는 내장형 프로세서 효율적인 캐쉬 구조임을 보여주는 것이다.

〈표 4〉 면적 비용 대 성능 비교

Caches	Capacity	Area	Average miss ratio (%)	Average AMAT (cycles)
		CACTI-III	MediaBench	
DM 16KB-32b	16KB	0.010953	1.76	1.26
VT 16KB-32b	16KB	0.012136	1.16	1.17
4W 8KB-32b	8KB	0.006590	1.51	1.23
4W 16KB-32b	16KB	0.011899	1.05	1.16
PSC-4KB-1KB-32b	5KB	0.006587	4.80	1.72
PSC-16KB-1KB-32b	17KB	0.013386	1.68	1.25
SSC-4KB-1KB-32b	5KB	0.006587	4.67	1.70
SSC-16KB-1KB-32b	17KB	0.013386	1.69	1.25
4KB8b-1KB32b	5KB	0.006179	0.88	1.13

〈표 5〉 캐쉬 접근 소모 전력

Caches	DM 16KB-32b	4W 16KB-32b	VT 16KB-32b		PSC-16KB-1KB-32b		4KB8b-1KB32b	
			DM	Victim	DM	Stream	DMC	FAB
Power(*nJ*)	0.27717	0.605081	0.27717	0.13436	0.22266	0.1496	0.223009	0.1607



(그림 8) MediaBench : 전력 소모 비교

5. 결 론

본 연구는 멀티미디어 응용을 위한 내장형 시스템에서 데이터 캐쉬의 성능 향상을 위한 새로운 캐쉬 시스템을 설계하는 것으로, 기본적으로 이중 캐쉬 구조 기반으로 용량을 작게 설계함으로 구현이 간단하고 저 전력과 저 비용에 중점을 두었다. 이러한 연구 목적을 달성하기 위하여 공간적 지역성을 효과적으로 이용할 수 있는 1KB 완전연관 버퍼(FAB)와 시간적 지역성의 효과를 극대화시키기 위해 작은 블록 크기와 선택적 저장 방식을 채택한 5KB 직접사상 캐쉬(DMC) 병행하여 전체 캐쉬 용량을 5KB로 작게 설계 하였다. 작은 용량에서 나올 수 있는 문제점은 적응적 다중 블록 선인출(AMBP)와 블록 필터링(EBF) 기능을 전체적인 시스템에 무리를 주지 않으면서 첨가하여 성능향상과 저전력을 꾀하였다.

실험 결과에 따르면 제안된 5KB 용량 캐쉬의 평균 접근 실패율과 평균 메모리 참조 시간은 이보다 3배의 용량을 가진 4-way 집합연관 캐쉬와 동등하였다. 또한 전력 소비적인 측면에서도 비교 캐쉬들에 비해 40% 이상의 감소 효과를 얻을 수 있었다. 이는 저전력 고성능을 요구하는 내장형 시스템의 구조에 적합한 캐쉬 구조라 할 수 있다.

참 고 문 헌

[1] W. Shiue, S. Udayanarayanan, and C. Chakrabati, "Data memory design and exploration for low-power embedded systems," ACM Trans. Design Automation of Electronic Systems, Vol.6 No.4, pp.553-568, Oct., 2001.

[2] S. Santhanam, "StrongARM SA110-A 160MHz 32b 0.5W CMOS ARM Processor," Hot Chips 8: A Symposium on High-Performance Chips, Aug., 1996.

[3] P. Struik, P. van der Wolf, and A. D. Pimentel. "A combined hardware/software solution for stream prefetching in multimedia applications," Proceedings of the SPIE Multimedia Hardware Architectures, pp.120-130, Jan., 1998.

[4] D. F. Zucker, R. B. Lee, M. J. Flynn, "Hardware and software cache prefetching techniques for MPEG benchmarks," IEEE Trans. Circuits and Systems for Video Technology, Vol.10, No.5, pp.782-796, Aug., 2000.

[5] R. Cucchiara, M. Piccardi, and A. Prati, "Neighbor cache prefetching for multimedia image and video processing," IEEE Trans. Multimedia, Vol.6, No.4, pp.539-552, Aug., 2004.

[6] I. Kuroda, and T. Nishitani, "Multimedia processors," Proc. the IEEE Vol.86, No.6, pp.1203-1221, June, 1998.

[7] Z. Xu, S. Sohoni, R. Min, and Y. Hu, "An analysis of cache performance of multimedia applications," IEEE Trans. Computers Vol.53, No.1, pp.20-38, Jan., 2004.

[8] P. Soderquist and M. Leeser, "Optimizing the data cache performance of a software MPEG-2 video decoder," Proceedings of the fifth ACM International Multimedia Conference, pp.291-301, Nov., 1997.

[9] C. Zhang, F. Vahid, and W. Najjar, "A highly configurable cache architecture for embedded systems," Proceedings of the 30th Annual International Symposium on Computer Architecture, pp.136-146, June, 2003.

[10] J. H. Lee, S. W. Jeong, S. D. Kim, and C. C. Weem, "An intelligent cache system with hardware prefetching for high performance," IEEE Transactions on Computers, Vol.52, No.5, pp.607-616, May, 2003.

[11] J. Tse and A. J. Smith, "CPU cache prefetching: Timing evaluation of hardware implementations," IEEE Trans. Computers Vol.47, No.5, pp.509-526, May, 1998.

[12] Z. Hu, S. Kaxiras, and M. Martonosi, "Timekeeping techniques for predicting and optimizing memory behavior," Proceedings of the IEEE International Solid-State Circuits Conference 2003, Digest of Technical Papers, Vol.1, pp.166-485, 2003.

[13] D. Burger and T. M. Austin, "The SimpleScalar tool set, version 2.0, Technical Report TR-97-1342," University of Wisconsin-Madison, 1997.

[14] J. Edler and M. D. Hill, "Dinero IV Trace-Driven Uniprocessor Cache Simulator," University of Wisconsin, http://www.cs. wisc.edu/~markhill/DineroIV.

[15] G. Reinman. and N. P. Jouppi, "CACTI 3.0: An integrated cache timing and power, and area model," Compaq WRL Report, Aug., 2001.

김 정 길

e-mail : cgkim@parallel.yonsei.ac.kr
2003년 연세대학교 컴퓨터과학과 (공학석사)
2003년~현재 연세대학교 컴퓨터과학과 박사과정
관심분야 : 컴퓨터 구조, 멀티미디어 내장형 시스템 등

김 신 덕

e-mail : sdkim@yonsei.ac.kr
1981년 연세대학교 전자공학과(공학사)
1987년 University of Oklahoma 전기컴퓨터공학과(공학석사)
1991년 Purdue University 전기컴퓨터공학과(공학박사)
1995년~현재 연세대학교 컴퓨터과학과 교수
관심분야 : 고성능 컴퓨터 구조, 지능형 캐쉬 메모리, 병렬처리 시스템, 그리드 컴퓨팅 등