지역성 결정 메커니즘을 기반으로 한 이중 캐쉬 시스템

(Dual Cache System based on the Locality Decision Mechanism)

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

요 약 캐쉬의 성능을 향상시키는 가장 효과적인 방법은 프로그램 수행 특성에 내재되어 있는 시간적 (temporal locality) -공간적 지역성 (spatial locality)을 활용하는 것이다. 본 논문에서는 추가적인 장치나 컴파일러의 도움 없이 단지 캐쉬의 구조적인 특징과 간단한 메커니즘만을 이용하여 두 가지 타입의 지역성을 효과적으로 반영할 수 있는 새로운 캐쉬 시스템이 제안된다. 제안하는 새로운 캐쉬 시스템은 다른 블록 크기와 다른 연관도를 가지는 두개의 캐쉬로써 구성되어 진다. 즉 작은 블록 크기를 지원하는 직접사상 캐쉬 (direct-mapped cache)와 큰 블록을 지원하는 완전 연관 버퍼 (fully-associative buffer)로 구성되어 진다. 큰 블록은 여러 개의 작은 블록으로 구성되어지며 두 캐쉬에서 접근 실패가 발생할 경우 직접사상 캐쉬의 접근 실패가 발생한 작은 블록과 그 이웃 작은 블록을 완전 연관 버퍼에 저장시킴으로써 한번 참조가 일어난 블록의 이웃 블록이 참조될 확률이 높다는 공간적 지역성의 특성을 효과적으로 반영할 수 있다. 또한 참조가 일어난 블록은 제어 비트를 사용하여 선택적으로 작은 블록을 직접사상 캐쉬에 저장함으로써 시간적 지역성을 보다 효과적으로 사용할 수 있다. 시뮬레이션 결과에 따르면 기존의 직접사상 캐쉬의 4배 크기보다도 좋은 성능 향상을 보이고 있으며, 동일한 크기의 victim 캐쉬보다 우수한 성능을 보이고 소비 전력 면에서는 5% 정도의 전력 감소를 보이고 있다.

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. In this paper, a new cache system is proposed to exploit both types of locality via simple mechanism without the aid of compiler or any additional component. The proposed cache system consists of two caches with different associativities and the different block sizes, i.e., a direct-mapped cache with small block size and a fully associative buffer with large block size as a multiple of small blocks. Therefore, spatial locality can be exploited by fetching large blocks including missed small block into the buffer, and temporal locality can be also exploited by selectively storing small blocks that were referenced at the buffer in the past. To determine the blocks to be stored at the direct-mapped cache, the proposed cache system uses one control bit. Therefore, more temporal locality can be managed by selectively keeping those blocks with high possibility of reference tendency in time domain. According to the simulation results, similar performance can be achieved by using four times smaller cache size comparing with the conventional direct-mapped cache. In addition, power consumption of the proposed cache can be reduced by about 5% comparing with the victim cache.

† 학생회원 : 연세대학교 컴퓨터과학과
ljh@kurene.yonsei.ac.kr
†† 비 회 원 : 연세대학교 컴퓨터과학과
jslee@kurene.yonsei.ac.kr
††† 정 회 원 : 연세대학교 컴퓨터과학과 교수
sdkim@kurene.yonsei.ac.kr
논문접수 : 2000년 3월 30일
심사완료 : 2000년 10월 10일

1. 서 론

오늘날 컴퓨터 시스템 내의 다양한 분야 중에서 캐쉬 메모리는 메모리 접근 지연시간 (memory access latency)과 소비 전력을 줄이고 전체 시스템의 성능 향상을 높이기 위한 가장 기본적이고 효과적인 방법중의

하나로써 제안되어 왔다. 이러한 캐쉬 메모리 연구들은 주로 접근 실패율을 줄이기 위한 방향으로 진행되어왔으며, 특히 최근에는 프로그램 수행 특성에 적합한 두 가지 지역성을 효과적으로 활용할 수 있는 캐쉬 구조에 대한 연구가 진행되고 있다. 시간적 지역성은 최근에 참조가 일어난 블록이 가까운 시간 내에 다시 참조될 확률이 높다는 것을 의미하며, 공간적 지역성은 참조가 일어난 블록의 이웃블록이 참조될 확률이 높다는 것을 의미한다 [1]. 그러나 이러한 두 가지 지역성은 캐쉬 크기가 고정되어 있을 경우 서로 상반되는 특징을 가지고 있다 [2]. 즉 캐쉬 블록 크기가 커질수록 공간적 지역성은 효과적으로 반영할 수 있지만 캐쉬 엔트리 수의 감소로 시간적 지역성에 대한 효과는 감소된다. 그러나 제안된 캐쉬 시스템은 이러한 지역성을 효과적으로 반영할 수 있는 새로운 캐쉬 시스템이다.

제안된 캐쉬 시스템의 성능 향상은 지역성의 기본적인 특성을 효과적으로 반영함으로써 얻을 수 있다. 즉 두 가지 지역성을 이용하기 위하여 작은 블록 크기를 가지는 직접사상 캐쉬와 큰 블록 크기를 가지는 완전 연관 버퍼로 구성되어 진다. 특히 작은 블록 크기는 시간적 지역성을 효과적으로 반영하기 위한 것이며 큰 블록 크기는 공간적 지역성을 반영하기 위한 캐쉬의 구조적인 특징이라고 할 수 있다. 또한 큰 블록은 여러 개의 작은 블록으로 구성되며 선입선출 (first in first out) 대체 알고리즘을 사용하고 있다. 직접사상 캐쉬와 완전 연관 버퍼에서 모두 접근 실패가 발생할 경우 항상 직접사상 캐쉬에서 접근 실패가 발생한 작은 블록을 포함한 큰 블록만이 완전 연관 버퍼에 저장되어지며 큰 블록내의 작은 블록들 중 참조가 일어난 작은 블록들 중에서 한번이라도 참조가 일어난 블록은 그 블록을 포함하는 큰 블록이 대체 (replacement)되어 버퍼에서 나가는 순간에 직접사상 캐쉬로 사상 (mapping) 되어진다. 이러한 버퍼의 엔트리 개수에 의한 시간 간격 메커니즘은 접근 실패 발생 시 직접사상 캐쉬에 바로 저장되는 메커니즘에 비해 시간적 지역성을 가진 블록들을 오래 동안 캐쉬 내에 존재시킬 수 있으며 충돌에 의한 접근 실패 (conflict miss)나 스래싱 효과 (thrashing effect)를 효과적으로 줄일 수 있는 장점을 가지고 있다.

내장형 프로세서에서 캐쉬 메모리의 전력 소비는 특히 중요한 요인으로써 고려되어 진다. [3]의 연구는 메모리 시스템의 하위 계층에 대한 이동 량을 줄일 수 있기 때문에 고성능 캐쉬가 전력 소비 면에서도 우수한 성능을 보이고 있음을 나타냈었으며, [4]의 연구는 일차 캐쉬 (L1)를 접근하기 전에 작은 필터 캐쉬 (filter cache)를 사용하여 전력 소비를 줄이고자 하는 연구를 수행하였다. 이에 제안된 캐쉬 구조는 접근 실패율을 줄임으로써 오프-칩 (off-chip) 전력 소비를 줄이고자 하였으며 또한 작은 캐쉬 크기와 두 가지 캐쉬가 모두 작은 블록 크기로 동작되기 때문에 온-칩 (on-chip)상의 전력 소비를 크게 줄일 수 있다.

시뮬레이션 결과에 따르면 추가적인 시간 간격에 의한 시간적 지역성의 효과로부터 충돌 접근 실패율은 대략 26%정도 줄일 수 있었으며, 완전 연관 버퍼에 의한 공간적 지역성의 효과로 약42%의 접근 실패율을 줄일 수 있었다. 따라서 제안된 캐쉬 시스템은 기존의 직접사상 캐쉬의 4배보다도 좋은 성능 향상을 보이고 있으며 victim 캐쉬보다도 약간 우수한 성능을 보이고 있다. 또한 전력 소비 측면에서는 victim 캐쉬에 비해 대략 5%의 감소 효과를 얻을 수 있었다.

이 논문의 나머지 부분은 다음과 같다. 관련 연구는 제 2장에서 소개되어지며, 제 3장은 L1 캐쉬의 성능을 향상시키기 위한 제안된 기술을 설명한다. 제 4장에서는 성능 평가 지표와 면적 대 성능 비교 그리고 소비 전력에 대한 시뮬레이션 결과를 비교-분석한다. 마지막으로 제 5 장에서 결론을 맺는다.

2. 관련 연구

Split temporal / spatial cache (STS) [5]은 선인출 (prefetch) 기법을 이용한 공간적 캐쉬 (spatial cache)와 L1, L2 계층구조를 가진 시간적 캐쉬 (temporal cache)로 구성되어진다. 시간적 캐쉬는 워드 단위의 블록 크기를 지원하고 있으며 공간적 캐쉬는 일반적인 블록 크기를 지원한다. 컴파일러에 의해 데이타들은 시간적 지역성과 공간적 지역성을 나타내는 데이타로 분류되어지며 메모리 참조 시 시간적 지역성을 가진 데이타는 시간적 캐쉬에 공간적 지역성을 가진 데이타는 공간적 캐쉬에 각각 저장되어진다.

Selective cache [6]은 큰 블록을 지원하는 공간적 캐쉬와 작은 블록을 지원하는 시간적 캐쉬 그리고 지역성 예측 테이블 (locality prediction table)로 구성되어진다. 만약 요청된 데이타가 두 캐쉬에 존재하지 않을 경우 접근 실패가 발생하게 되며 프로세서는 접근 실패를 처리할 동안 지연되게 된다. 이때 요청된 데이타는 지역성 예측 테이블의 정보로부터 판단되어지며 시간적 지역성을 가진 데이타로 판단되어지면 시간적 캐쉬에 저장되고 공간적 지역성을 가진 데이타로 판단되면 공간적 캐쉬에 저장되게 된다. 그러나 어떤 지역성도 가지지 않는다고 판단될 경우 이러한 데이타는 캐쉬에 저장

되지 못하며 지역성 예측 테이블에 그 정보만을 저장시키게 된다.

HP-7200 assist cache [7]은 직접사상 캐쉬와 같은 계층의 완전 연관 캐쉬로 구성된다. 메모리 참조 시 두 캐쉬를 동시에 접근하며 한 사이클 동안 접근 적중/실패를 결정하게 된다. 접근 실패가 발생하거나 선인출 메커니즘을 사용하여 블록을 버퍼에 저장하게 되며 시간적 지역성을 가진 데이타로 판단될 경우에만 직접사상 캐쉬에 저장하는 방식을 사용하였다.

Bennett and Flynn [8]이 제안한 ***prediction caches***는 최근에 캐쉬 접근 실패의 기록 (history)을 저장하여 앞으로의 접근 실패를 예측하는 캐쉬 시스템이다. 이 시스템은 선인출 기법과 victim 캐쉬 기법을 혼합한 형태라 할 수 있다.

Rivers and Davidson [9]이 제안한 ***NTS cache***는 완전 연관 버퍼와 직접사상 캐쉬로 구성되며 시간적 지역성과 비시간적 지역성 (non-temporal)을 가진 블록을 구분하여 시간적 지역성을 가진 데이타를 효과적으로 사용하고자 하는 방식을 사용하였다. 이 구조의 가장 큰 특징은 캐쉬의 오염을 (cache pollution)을 효과적으로 줄일 수 있다.

위에서 제시된 다양한 캐쉬 구조와 제안된 캐쉬와의 두드러진 차이점은 다음과 같다. 제안된 캐쉬의 구조적인 특징은 다른 연관도 (associativity)와 다른 블록 크기를 지원하는 것이다. 그러나 STS 캐쉬와 selective 캐쉬는 다른 블록 크기를 지원하지만 같은 연관도로 구성된 캐쉬 구조이며, victim 캐쉬와 assist 캐쉬 그리고 NTS 캐쉬 시스템은 같은 블록 크기를 지원하지만 다른 연관도를 가진 캐쉬 구조이다. 또한 캐쉬의 성능을 향상시키기 위한 메커니즘으로는 서로 상이한 특성을 가지고 있다. 먼저 STS 캐쉬는 공간적 지역성을 이용하기 위하여 선인출 기법을 사용하고 있으며 selective 캐쉬는 요청한 데이타가 어떤 지역성을 가지는지를 판단하기 위한 지역성 예측 테이블을 활용하고 있다. 그리고 NTS 캐쉬는 시간적 지역성을 효과적으로 이용하기 위한 구조이며 추가적으로 시간적 지역성을 예측하는 장치를 가지고 있다. 그러므로 이러한 세 가지 캐쉬 시스템은 지역성을 결정하는 방법과 한가지 지역성만을 효과적으로 반영하기 위한 구조라 할 수 있다. 그러나 제안된 캐쉬 시스템은 시간 간격을 바탕으로 한 간단한 하드웨어 제어 메커니즘만을 이용하고있으며 다른 블록과 다른 연관도를 바탕으로 한 구조적인 특징만으로 두 가지 지역성을 효과적으로 이용할 수 있는 새로운 캐쉬 시스템이다. 마지막으로 HP-7200 assist 캐쉬는 시간 간격과 선인출 기법을 사용하여 시간적 지역성을 효과적으로 이용하고자 한 캐쉬 시스템으로 시간적 지역성을 결정하기 위한 정보는 컴파일러의 지원을 필요로 하였다 .

3. 제안된 캐쉬의 구조적 특징과 동작 원리

이 장에서는 제안된 캐쉬의 동작 모델과 구조적 특징을 설명하고 새로운 캐쉬 시스템을 제안하게 된 배경과 동기에 대해서 상세히 살펴 볼 것이다.

3.1 캐쉬 설계 동기 및 방법

새로운 캐쉬의 주목적은 빠른 접근 시간 (access time), 다양한 응용 프로그램에 적합한 시간적-공간적 지역성을 이용한 접근 실패 최소화, 그리고 저 전력 소비에 중점을 두고 설계하고자 하였다. 그러나 단일 캐쉬 구조만으로는 상반된 특성을 가진 두 가지 지역성을 효과적으로 이용하는데 한계점이 있으므로 각각의 지역성을 효과적으로 반영할 수 있는 두 개의 분리된 캐쉬 구조를 선택하게 되었다.

두개의 분리된 캐쉬 중 작은 블록 크기 가진 직접사상 캐쉬는 저 전력 소비와 빠른 접근 시간에 가장 적합한 구조임으로 주 캐쉬로써 선택하였다. 그러나 직접사상 캐쉬 자체만으로는 많은 단점을 내포하고 있기 때문에 이러한 단점을 보상하기 위하여 큰 블록을 가진 완전 연관 버퍼를 같은 계층에 위치시켰다. 또한 직접사상 캐쉬의 작은 블록 크기는 엔트리 수 증가에 의한 시간적 지역성을 반영하기 위한 구조이며, 완전 연관 버퍼의 큰 블록 크기는 공간적 지역성을 반영하기 위한 구조이다. 그리고 시간적 지역성이 높은 특정한 작은 블록을 선택적으로 직접사상 캐쉬에 저장시킴으로써 하나의 저장 공간 (one set)만을 가진 직접사상 캐쉬의 효용성을 증가시킬 수 있다. 즉 완전 연관 버퍼의 큰 블록은 직접사상 캐쉬가 지원하는 작은 블록 크기와 동일한 블록들로써 구성되며 버퍼에서 대치될 때까지 작은 블록들은 버퍼 내에 존재하게 된다. 그리고 큰 블록 내에 속하는 작은 블록들은 버퍼에서 대치되는 순간 과거에 한번이라도 참조가 일어난 블록만 선택하여 직접사상 캐쉬로 저장함으로 이러한 방식의 효과는 직접사상 캐쉬에 바로 저장시키는 메커니즘에 비해 충돌에 의한 접근 실패와 스래싱 효과를 줄일 수 있고 시간적 지역성을 가진 데이타의 수명 시간을 연장시킬 수 있는 장점을 가지게 된다.

완전 연관 버퍼의 큰 블록에 대한 효과는 다음과 같다. 두 캐쉬에서 접근 실패가 발생한 경우 직접사상 캐쉬에서 접근 실패가 발생한 작은 블록을 포함하는 여러

개의 이웃 블록들을 미리 인출함으로써 공간적 지역성을 효과적으로 반영할 수 있다 (예로 작은 블록의 4배 또는 작은 블록의 8배). 시뮬레이션 결과에 따르면 이웃 블록을 인출할 경우 대부분의 벤치마크 (benchmark)에서 이웃 블록들이 접근 성공할 확률이 50% 이상인 것으로 나타났다. 이러한 확률은 공간적 지역성에 대한 효과가 대단히 높음을 알 수 있다. 이는 기존의 일반적인 직접사상 캐쉬에서 8-byte 블록 크기를 가지는 캐쉬보다 32-byte 블록 크기를 가진 캐쉬가 낮은 접근 실패율을 보이는 것은 작은 블록 크기에 의한 엔트리 수 증가로부터 얻을 수 있는 시간적 지역성 효과보다 큰 블록에 의한 공간적 지역성에 의한 효과가 더 높기 때문이다. 그러므로 버퍼의 블록 크기는 클수록 좋은 성능을 보이지만 전송 시간과 접근 실패에 대한 타협 (tradeoff)이 존재하게 된다.

이러한 접근 방법을 바탕으로 제안된 캐쉬의 구조적 특징과 캐쉬 메커니즘을 이용하여 성능 향상을 얻을 수 있다. 그림 1은 직접사상 캐쉬에 대한 엔트리 개수의 증가 없이 단지 버퍼의 추가로부터 얻을 수 있는 추가적인 시간 간격에 의한 충돌 접근 실패 감소와 큰 블록 크기에 의한 공간적 지역성에 의한 접근 실패 감소율을 보이고 있다. 그림 1에서 성능 향상은 기존의 직접사상 캐쉬에서 일어나는 접근 실패에 대해서 큰 블록의 버퍼를 사용하였을 때의 효과를 보여주며 접근 실패 감소에 대한 평균값으로 나타내었다. 즉 기존의 직접사상 캐쉬의 캐쉬 크기는 8KB이며 블록 크기는 8-byte이다. 그리고 제안된 캐쉬 구조는 직접사상 캐쉬가 8KB, 블록 크기는 8-byte이며 버퍼의 크기는 1KB, 블록 크기는 32-byte를 사용하였다. 그러므로 제안된 캐쉬로부터 직접사상 캐쉬의 엔트리 수에 의한 시간적 지역성은 반영되지 못하지만 버퍼로부터 시간 간격 (time interval)에 의한 시간적 지역성 (예로 충돌 접근 실패 감소)과 큰 블록에 의한 공간적 지역성에 의한 효과를 얻을 수 있다. 시뮬레이션의 결과 제안된 캐쉬의 직접 사상 캐쉬의 블록 크기가 기존의 직접사상 캐쉬의 블록 크기와 동일한 경우 버퍼에 의한 공간적 지역성에 대한 효과가 대단히 높음을 알 수 있다. 그림 2는 직접사상 캐쉬의 엔트리 증가에 의한 효과와 버퍼에 의한 시간 간격에 의한 효과를 나타내고 있다. 이들은 모두 시간적 지역성에 의한 효과라 할 수 있다. 그림 2에서는 기존의 직접사상 캐쉬의 캐쉬 크기는 8KB, 블록 크기는 32-byte이며 제안된 캐쉬의 직접사상 캐쉬는 그림1과 동일하다. 그러므로 제안된 캐쉬의 직접사상 캐쉬의 엔트리 수가 (32-byte/8-byte)배 증가함으로 시간적 지역성을 효과적으로 이용할 수 있으며 또한 버퍼에 의한 시간 간격까지 활용 할 수 있다. 그러나 기존의 직접사상 캐쉬도 32-byte, 제안된 캐쉬도 32-byte로 인출하기 때문에 공간적 지역성에 의한 효과는 얻을 수 없다. 그러나 접근 실패 감소율은 그림 1과 마찬가지로 대단히 높음을 알 수 있다. 즉 기존의 직접사상 캐쉬가 어떤 블록 크기를 가지더라도 제안된 캐쉬는 기존의 캐쉬에 비해 효과적으로 접근 실패율을 줄일 수 있다.

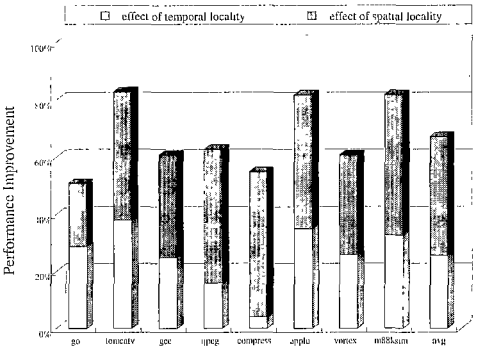

그림 1 기존의 직접사상 캐쉬와 제안된 캐쉬의 직접사상 캐쉬의 블록 크기가 같은 경우 시간적 지역성과 공간적 지역성에 의한 기존의 캐쉬 접근 실패 감소율

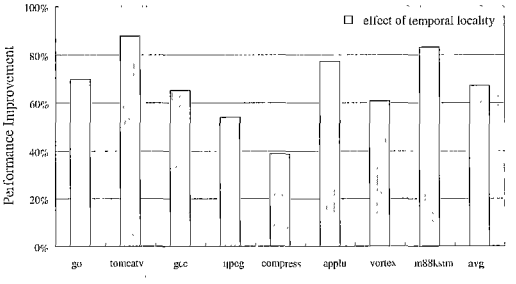

그림 2 기존의 직접사상 캐쉬와 제안된 캐쉬의 완전 연관 버퍼의 블록 크기가 같은 경우 제안된 캐쉬의 시간적 지역성에 의한 기존의 접근 실패 감소율

3.2 제안된 캐쉬의 구조

이 연구의 주목적은 간단하지만 저 전력과 높은 성능 향상을 보이는 캐쉬 시스템을 설계하는 것이다. 따라서 주어진 캐쉬 공간을 최적화 시키는 방법으로 시간적-공간적 지역성을 효과적으로 이용할 수 있는 두 개의 캐쉬로 구성된 캐쉬 시스템을 제안하고자 한다. 제안된 캐

쉬 시스템은 기존의 직접사상 캐쉬와 같은 계층의 완전 연관 버퍼로써 구성되어 진다.

제안된 캐쉬에서 직접사상 캐쉬는 n-byte 블록 크기를 가지는 m 개의 작은 블록으로 구성되어지며 완전 연관 버퍼는 여러 개의 작은 블록으로 구성되어진다. 즉 선택된 상수 c로부터 $c \times n$ byte, k개의 큰 블록으로써 구성되어 진다. 따라서 완전 연관 버퍼의 총 바이트는 $k \times c \times n$ byte 이다.

제안된 캐쉬의 구조는 그림 3과 같다. 직접사상 캐쉬는 주 캐쉬의 역할을 담당하며 일반적인 직접사상 캐쉬와 동일한 구조로써 구현되어진다. 완전 연관 버퍼의 데이타 저장 공간은 여러 개의 뱅크 (banks)로써 구성되어지며 각 뱅크의 블록 크기는 직접사상 캐쉬의 블록 크기와 동일하다. 그리고 버퍼의 태그 부분은 모든 엔트리를 동시에 비교할 수 있는 CAM (content addressable memory)으로 구현되어지며 각 뱅크에 저장되어지는 블록들은 한 비트를 가지고 있다. 이러한 비트는 큰 블록내의 작은 블록들 중에서 참조가 일어난 블록을 나타내기 위한 비트로 이후부터 적중 비트라 명명할 것이다.

CPU로부터 어떤 메모리 참조가 발생하게 되면 직접사상 캐쉬와 완전 연관 버퍼가 같은 계층에서 동시에 접근이 일어난다. 만약 직접사상 캐쉬에서 적중이 일어나면 기존의 직접사상 캐쉬처럼 동일한 방법으로 CPU 요청을 수행하게 된다. 만약 직접사상 캐쉬에서 접근 실패이고 완전 연관 버퍼에서 적중이 일어나면 큰 블록내의 부합되는 하나의 작은 블록만이 인출되어짐과 동시에 참조가 일어난 블록임을 나타내기 위하여 그 작은 블록의 적중 비트는 1로 설정되어 진다. 위에서도 언급했듯이 버퍼의 데이타 부분은 뱅크로 나누어져 있기 때문에 디코더 (decoder)를 이용하여 항상 버퍼 참조 시

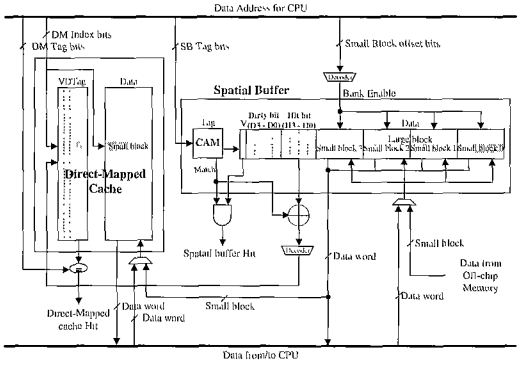

그림 3 제안된 데이타 캐쉬 구조

하나의 뱅크만 선택적으로 허가 (enable)가 가능하다. 이것은 소비 전력을 줄이기 위한 방법이며 두 가지 캐쉬 모두 작은 블록 크기처럼 동작되어 질 수 있다.

3.3 캐쉬 제어 알고리즘

여기서는 제안된 캐쉬 시스템의 동작 원리를 상세히 설명하고자 한다. CPU로부터 메모리 참조가 발생하게 되면 항상 직접사상 캐쉬와 완전 연관 버퍼가 동시에 참조가 일어난다. 가능한 경우로는 다음과 같다:

㉠ **직접사상 캐쉬에서의 적중:** 직접사상 캐쉬에서 적중이 발생되어지면 일반적인 직접사상 캐쉬와 동일한 동작을 수행하게 된다. 읽기 명령어인 경우 요청한 데이타를 CPU로 보내고 캐쉬의 동작은 끝나게 되며, 쓰기 명령어인 경우 해당 블록에 쓰기 동작을 수행함과 동시에 갱신 비트 (dirty bit)을 1로 설정하게 된다.

㉡ **완전 연관 버퍼에서의 적중:** 메모리 주소가 CPU로부터 발생되어질 때 주소의 일부 비트를 이용하여 버퍼 내 여러 개의 뱅크들 중에서 하나의 뱅크만 선택하여 사용할 수 있다. 만약 설계의 예처럼 작은 블록 크기가 8-byte이고 큰 블록 크기가 32-byte인 경우 뱅크의 수는 그림 3처럼 4개로써 구성되어진다. 따라서 한번에 하나의 뱅크만 선택되어짐으로 전력 소비적인 측면에서 유리한 장점을 가지게 된다. 또한 각 뱅크의 블록들은 참조가 일어난 블록과 참조가 일어나지 못한 블록을 구별하기 위하여 한 비트의 제어 비트를 가지게 된다. 만약 완전 연관 버퍼내의 하나의 뱅크에서 적중이 발생되면 읽기 명령인 경우 적중이 발생한 뱅크의 작은 블록으로부터 요청한 데이타를 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 직접사상 캐쉬와 완전 연관 버퍼에서 모두 접근 실패가 발생하면 직접사상 캐쉬에서 접근 실패가 발생한 작은 블록을 포함한 큰 블록만이 다음 메모리 계층으로부터 인출되어 완전 연관 버퍼에 저장되어진다. 가령 그림 3에서처럼 작은 블록이 8-byte이고 큰 블록이 32-byte인 경우 32-byte 블록의 경계 (boundary)에 속하는 4개의 순차적인 작은 블록이 인출되어 버퍼에 저장되어지며 완전 연관 버퍼의 유효 비트 (valid bit)가 모두 1인 경우와 그렇지 않은 경우로 나누어진다.

- 완전 연관 버퍼의 유효 비트가 모두 1인 아닌 경우 (fully-associative buffer is not filled up):

접근 실패가 발생하였을 때 버퍼 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 데이타를 CPU로 보내고, 쓰기 접근 실패이면 쓰기 정정 (write-back) 모드 방식으로 동작하게 되며 요청한 큰 블록을 버퍼에 저장시키게 된다.

- 완전 연관 버퍼의 유효 비트가 모두 1인 경우 (fully-associative buffer is filled up):

완전 연관 버퍼의 대체 알고리즘으로 선입선출 방식을 채택하였다. 만약 완전 연관 버퍼의 모든 엔트리가 유효 상태이면 접근 실패 시 가장 오래된 큰 블록이 선택되어지며 그 대체 블록에 속하는 작은 블록들 중에서 이미 참조가 일어난 블록을 선택하여 직접사상 캐쉬로 저장하게 된다. 그러므로 완전 연관 버퍼에서 직접 다음 계층 메모리로 쓰기 정정되는 경우는 발생할 수 없다. 직접사상 캐쉬 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.

제안된 캐쉬의 동작 흐름도 (flowchart)는 그림 4와 같다.

4. 시뮬레이션과 분석적 모델을 통한 성능 평가

이 장에서는 시뮬레이션 수행을 통한 접근 실패율과 평균 메모리 접근 시간을 비교 분석하고 면적 대 성능비 그리고 전력 소비에 대해서 분석적 모델을 사용하여 설명하고자 한다. 시뮬레이션 수행을 위한 벤치마크로는 SPECint95에서 6개, SPECfp95에서 2개 (예로 applu와 tomcatv)를 사용하였으며 트레이스 구동 (trace-driven) 시뮬레이션을 수행하였다. 또한 트레이스를 생성하기 위해서 QPT2[10]를 이용하였다. 그리고 생성된 트레이스를 이용하여 제안한 캐쉬를 시뮬레이션하기 위해 DineroIV 시뮬레이터를 수정하였다. 비교 대상으로는 일반적인 기존의 직접사상 캐쉬와 다양한 캐쉬 구조들 중 비교 분석 결과 가장 높은 성능 향상을 보였던 victim 캐쉬를 선택하였다.

4.1 직접사상 캐쉬 대 제안된 캐쉬의 성능 비교 분석

4.1.1 접근 실패율

제안된 캐쉬의 직접사상 캐쉬와 완전 연관 버퍼의 최적의 블록 크기를 결정하기 위하여 다양한 시뮬레이션을 수행하였다. 분석적 모델과 시뮬레이션 결과 직접사상 캐쉬의 블록은 가능한 작을수록 (예로 4-byte) 큰 블록 크기는 가능한 한 클수록 (64-byte or 128byte) 좋은 접근 실패율을 보이고 있다. 그러나 전송 시간과 하드웨어 비용 그리고 전력 소비 등 다양한 요인을 감안할 때 최적의 블록 크기는 작은 블록 크기로 8-byte, 큰 블록 크기로 32-byte를 최적의 블록 크기로 결정하였다.

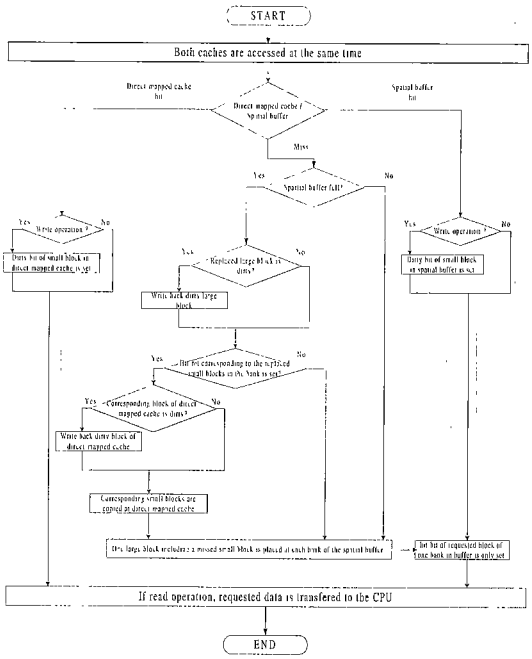

그림 4 제안된 캐쉬 구조의 동작 흐름도

기존의 일반적인 직접사상 캐쉬와 제안된 캐쉬와의 접근 실패율을 그림 5에서 보여주고 있다. 그림에서 기존의 직접사상 캐쉬는 DC로 표기하였으며 "16k-16byte"는 16-byte의 블록 크기를 가진 16KB 캐쉬 크기를 나타낸다. 그리고 제안된 캐쉬의 "8k8byte-1k32byte"는 8-byte 블록크기를 가진 8KB 직접사상 캐쉬와 32-byte 블록 크기를 가진 1KB 완전 연관 버퍼를 나타내고 있다. 시뮬레이션은 각각의 벤치마크에 대해 다양한 블록 크기와 캐쉬 크기를 변화시켜가면서 수행하였으며 결과적으로 제안된 캐쉬는 4배의 일반적인 직접사상 캐쉬와 거의 유사한 성능 효과를 보이고 있다. 또한 제안된 캐쉬에서 직접사상 캐쉬 크기를 증가시키

는 것보다 완전 연관 버퍼 크기를 증가시키는 것이 더 효과적인 성능 향상을 보이고 있음을 알 수 있다.

그림 6은 2-way 집합 연관 캐쉬와 제안된 캐쉬와의 접근 실패율을 보여주고 있다. 그림에서도 알 수 있듯이 2-way 집합 연관 캐쉬는 직접 사상 캐쉬에 비해 접근 실패율을 크게 줄이는 장점을 가지는 구조이지만 느린 접근 시간과 높은 전력 소비로 내장형 프로세서나 이동형 단말기와 같은 저 전력을 필요로 하는 시스템에서는 거의 사용할 수 없는 단점을 가지고 있다.

시뮬레이션 결과에서도 알 수 있듯이 제안된 캐쉬는 2배 크기의 2-way 집합 연관 캐쉬와 거의 비슷한 성능을 보이고 있으며, 소비 전력까지 고려한다면 제안된 캐쉬 시스템이 더욱 효율적인 캐쉬 시스템임을 알 수 있을 것이다.

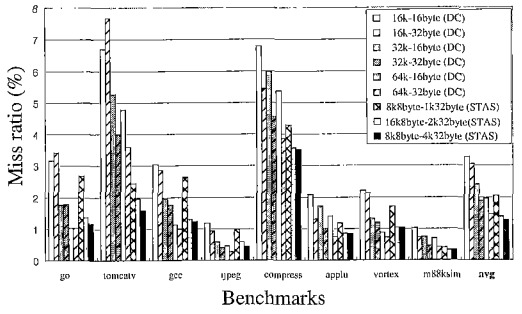

그림 5 기존의 직접사상 캐쉬와 제안된 캐쉬의 접근 실패율

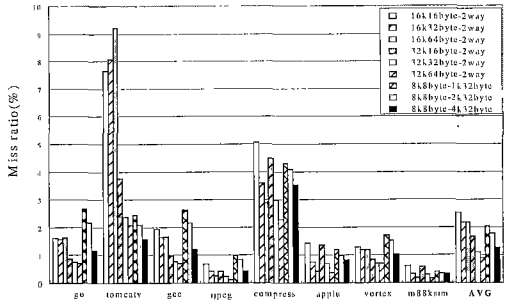

그림 6 기존의 2-way 집합 연관 캐쉬와 제안된 캐쉬의 접근 실패율

4.1.2 평균 메모리 접근 시간

캐쉬 시스템의 성능 평가를 보다 정확하게 측정하기 위한 방법으로 평균 메모리 접근 시간을 이용하였다. 평균 메모리 접근 시간을 얻기 위한 식은 다음과 같다.

*Average memory access time = Hit time + Miss rate * Miss penalty.* (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 1로써 정의되어진다. 이러한 변수 값들은 일반적인 32-bit 내장형 프로세서에서 사용되어지는 값들을 사용하였다 (예로 Hitachi SH4 or ARM920T).

표 1 시뮬레이션 변수들

System parameters	Values
CPU clock	200 MHz
memory latency	15 /cpucycle
memory bandwidth	1.6 Gbytes / sec
direct-mapped cache hit time	1 /cpucycle
fully-associative buffer hit time	1 /cpucycle

일반적인 직접사상 캐쉬와 제안된 캐쉬와의 평균 메모리 접근 시간에 대한 시뮬레이션 결과는 그림 7과 같다. 그림 5의 접근 실패율과 큰 차이는 없으나 블록이 커질수록 전송 시간이 길어지기 때문에 접근 실패율에서 좋은 성능을 보이던 것이 도리어 평균 메모리 접근 시간에서는 낮은 성능을 보일 수 있다. 벤치마크 분석 결과에 따르면 tomcatv와 같이 지역성이 대단히 높은 응용프로그램에 대해서 특히 제안된 캐쉬가 높은 성능을 보임을 알 수 있었다.

4.2 Victim 캐쉬 대 제안된 캐쉬의 성능 비교 분석

기존의 일차 (L1) 캐쉬의 성능 향상을 높이기 위해 제시된 다양한 방법들 중에서 시뮬레이션 분석 결과 victim 캐쉬가 가장 우수한 성능을 보임을 알 수 있었다. Victim 캐쉬는 직접사상 캐쉬의 가장 큰 단점인 충돌에 의한 접근 실패를 효과적으로 줄임으로써 시스템 전체 접근 실패율을 낮출 수 있었다. 그러나 주 캐쉬인 직접사상 캐쉬와 victim 버퍼 사이에 많은 데이타 교체 (swapping)가 일어나며 높은 성능을 얻기 위해서 큰 블록 크기를 제공해야만 하기 때문에 전력 소비를 증가시키는 요인이 된다.

Victim 캐쉬의 동작 모델은 다음과 같다. CPU로부터 메모리 참조 주소가 발생하면 제안된 캐쉬처럼 메인 캐쉬와 victim 버퍼가 동시에 참조되어진다. 만약 victim 버퍼에서 적중이 발생하면 그 데이타는 CPU로 보내거나 쓰기 동작이 일어나며 동시에 주 캐쉬인 직접사상 캐쉬와 블록 교체가 일어난다. 이러한 동작이 완료되기 위해서는 추가적인 한-사이클이 필요 되어진다. 그리고

만약 두 캐쉬에서 모두 접근 실패가 발생할 경우 다음 메모리 계층으로부터 필요한 블록을 인출하여 직접사상 캐쉬에 저장하게 되며 동시에 직접사상 캐쉬의 대치 블록은 victim 버퍼로 저장되게 된다. 그러나 이러한 대치 동작은 캐쉬 제어기가 접근 실패를 수행하는 동안 처리할 수 있으므로 추가적인 시간 지연은 발생하지 않는다. 다만 전력 소비적인 측면에서는 이러한 동작 또한 고려되어져야 한다.

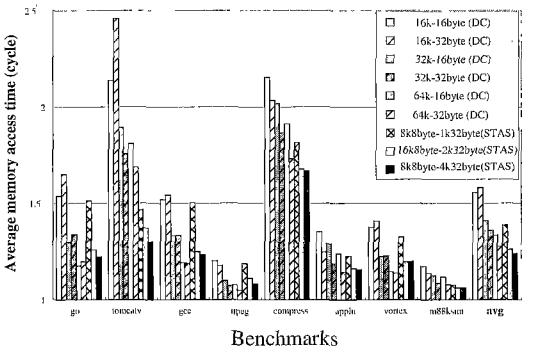

그림 7 기존의 직접사상 캐쉬와 제안된 캐쉬의 평균 메모리 접근 시간

그림 8과 그림 9는 동일한 캐쉬 크기를 가진 제안된 캐쉬와 victim 캐쉬와의 접근 실패율과 평균 메모리 접근 시간을 보여주고 있다. 제안된 캐쉬 구조는 8-byte 블록을 가진 8KB 직접사상 캐쉬와 32-byte 블록을 가진 완전 연관 버퍼로 구성되어지며, victim 캐쉬 또한 동일한 캐쉬 크기를 가진 직접사상 캐쉬와 victim 버퍼로써 구성되어 진다.

시뮬레이션 결과에 따르면 victim 캐쉬의 경우 블록 크기가 32-byte일 때 최고의 성능 향상을 보이고 있으나 블록 크기의 증가로 전력 소비 또한 증가되어 진다.

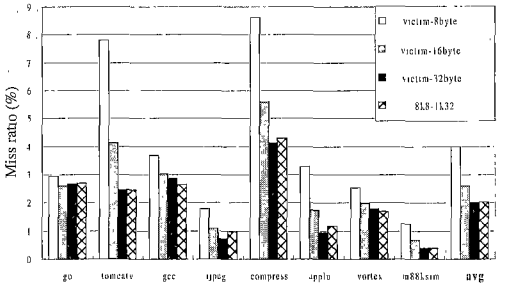

그림 8 동일한 캐쉬 크기를 가진 제안된 캐쉬와 victim 캐쉬와의 접근 실패율

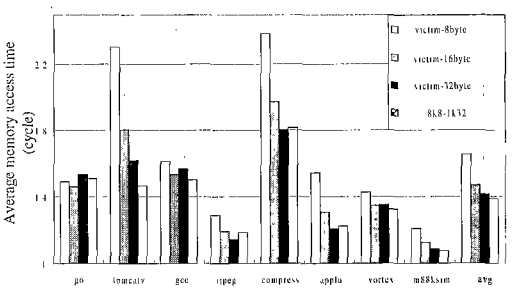

그림 9 동일한 캐쉬 크기를 가진 제안된 캐쉬와 victim 캐쉬와의 평균 메모리 접근 시간

그러나 제안된 캐쉬 시스템은 작은 블록과 뱅크에 의한 큰 블록 모두 8-byte 블록으로 동작되어지기 때문에 전력 소비 측면에서 더 유리한 장점을 가지게 된다. 그리고 그림 9에서 알 수 있듯이 성능 적인 측면에서도 다수 우위의 성능 향상을 보이고 있다.

4.3 면적 비용 대 성능 비에 대한 비교 분석

오늘날 완전 연관 버퍼의 태그 부분에 대한 회로는 모든 엔트리를 동시에 비교 할 수 있는 CAM으로 구성되어진다. 이러한 CAM 셀은 저장과 비교기를 모든 엔트리마다 가지고 있기 때문에 일반적으로 RAM 셀에 비해 두 배의 면적 크기로 구성되어진다 [11]. 이 장에서는 다양한 캐쉬 크기를 가진 직접사상 캐쉬와 제안된 캐쉬와의 비용 대 성능 비를 분석적 모델과 시뮬레이션을 이용하여 설명하고자 한다. 단위는 *rbe* (register bit equivalents)이며 전체 면적은 다음 식으로부터 얻을 수 있다:

$$Area = PLA + RAM + CAM. \quad (2)$$

여기서 제어 회로인 *PLA* (programmable logic array)는130 *rbe*, *RAM* 셀 (cell)은 0.6 *rbe*, 그리고 *CAM* 셀은 1.2 *rbe*로 가정하였다. 식 3은 *RAM* 면적을 나타낸다 [11]:

$$RAM = 0.6*(\#entries + \#Lsense\_amp)*((\#data\_bits + \#status\_bits) + Wdriver), \quad (3)$$

여기서 *Lsense_amp*는 비트-라인 센서 증폭기 (bit-line sense amplifier)의 비트 길이를 나타내며 *Wdriver*는 드라이버 (driver)의 데이타 폭 (width)을 나타내는 변수이다. 그리고 *#entries*는 태그 부분 또는 데이타 부분의 열 (row)의 수를 나타내며 *#data_bits*는 한 셋 (set)의 태그 비트 수 또는 데이타 비트 수를 나타낸다. 마지막으로 *#status_bits*는 한 셋의 상태 비트 수를 나타낸다. 그리고 식 4를 이용하여 *CAM*의 면적을 산출할 수 있다:

$CAM = 0.6*(2 * \#entries + \#Lsense\_amp) * (2 * \#tag\_bits + Wdriver)$, (4)

여기서 *CAM*의 사용은 완전 연관 버퍼의 태그 부분뿐이며 변수들은 식 3과 같다.

이러한 수식과 시뮬레이션을 바탕으로 다양한 캐쉬 크기를 이용한 성능/비용을 산출할 수 있다. 표 2에서 일반적인 직접사상 캐쉬의 블록 크기를 32-byte로 선택한 이유는 다른 블록 크기를 가진 경우보다 높은 성능 향상을 보이기 때문이다. 결과적으로 제안된 캐쉬는 캐쉬 크기가 4배 큰 직접사상 캐쉬보다도 오히려 좋은 성능을 보이고 있으며 제안된 캐쉬의 직접사상 캐쉬 크기를 증가시키는 것 보다 버퍼 크기를 증가시키는 것이 비용과 성능 적인 측면에서 좋은 결과를 보이고 있음을 알 수 있다.

4.4 전력 소비에 대한 분석적 모델과 비교 분석

CMOS 회로의 전력 소비는 게이트 정전용량 (gate capacitance)의 충전과 방전에 의해서 일어난다. 전체 소비 전력에 관한 식은 다음과 같다.

표 2 제안된 캐쉬와 직접사상 캐쉬의 성능 대 비용

Caches	Area (*rbe*)	Avg of Miss ratios (%)	Avg of average memory access times (%)
16KB-32byte (DC)	89640	3.0591	1.5812
32KB-32byte (DC)	177496	1.8859	1.3583
64KB-32byte (DC)	**352596**	**1.4563**	**1.2767**
8KB8byte-1KB32byte	67431	2.0430	1.3882
8KB8byte-2KB32byte	73680	1.7764	1.3375
8KB8byte-4KB32byte	**86178**	**1.2637**	**1.2401**
16KB8byte-1KB32byte	125795	1.5578	1.2960
16KB8byte-2KB32byte	132044	1.3708	1.2605

$Et = 0.5 * Ceq * V^2$ (5)

등가 정전용량 *Ceq*은 Wilton 와 Jouppi [12]가 제안한 분석적 모델을 이용하여 얻을 수 있다. 그들이 제안한 모델은 0.8 m 프로세스와 전압 3.3 V를 가정하였으며, *Ceq*는 제시되는 다양한 식으로부터 얻을 수 있다. 또한 각각의 트랜지스터에서 일어나는 변이 개수를 얻기 위해서 Kamble 과 Ghost [13, 14]가 수행한 연구를 참조하여 제안된 캐쉬에 적용하였다. [13, 14]에 의하면 전력 소비는 크게 4가지 구성요소, 즉 *Ebits*, *Eword*, *Eoutput*, 그리고 *Einput*로 나눌 수 있으며 각각은 비트-라인 (bit-lines), 워드-라인 (word-lines), 주소와 데이타 출력-라인 (output-lines), 그리고 주소 입력-라인 (input-lines)에서의 전력 소비를 나타낸다. 전체 캐쉬의 전력 소비는 다음 식과 같다:

$Ecache = Ebits + Eword + Eoutput + Einput.$ (6)

- **비트-라인에서의 전력 소비**:

*Ebits*는 SRAM이 참조되어질 때 모든 비트 라인에서 소비되는 전력을 나타낸다. 즉 데이타를 읽거나 쓰고자 할 때 블록의 정전용량 충전 (precharging)에 의해서 소비되는 전력이다. 직접사상 캐쉬의 경우 태그 부분과 데이타 부분이 동시에 참조되어지며 완전 연관 버퍼는 전력 소비를 최소화하기 위하여 먼저 태그 부분만 참조되어지고 적중이 발생한 경우에만 데이타 부분의 참조가 이루어진다. *Ebits*는 다음 식으로부터 구할 수 있다:

$Ebits = 0.5 * V2 * [Nbp*Cbp + M * (Nhit + Nmiss) * (8*L + T + C) * (Cg,Qpa + Cg,Qpb + Cg,Qp) + Nbw * Cba + Nbr * Cba]$, (7)

여기서 *Nbp*, *Nbw*, 그리고 *Nbr*는 각각 비트 라인 충전에 의한 변이의 개수, 쓰기 동작의 개수, 그리고 읽기 동작의 개수를 나타내는 변수 값이다. 그리고 *M*은 연관도, *L*은 한 블록 크기에 대한 바이트, *T*는 한 셋 (set)의 블록 당 태그 크기, 그리고 *C*는 블록 당 상태 비트의 수를 나타내는 변수들이다. *Cg,Qpa*, *Cg,Qpb*, 그리고 *Cg,Qp*는 트랜지스터 *CQ* 게이트 정전용량을 나타낸다. 마지막으로 *Cbp* 그리고 *Cba*는 쓰기/읽기 동작을 수행하기 위해 충전되는 각 비트 라인의 실효 부하 정전용량을 나타내고 있다. 식 8, 9는 [12]를 참조하여 얻을 수 있다:

$Cbp = Nrows * (0.5 * Cdrain,Q1 + Cbitwire).$ (8)

$Cba = Nrows * (0.5 * Cdrain,Q1 + Cbitwire) + Cdrain,Qp + Cdrain,Qpa.$ (9)

여기서 *Cdrain,Q1*는 트랜지스터 *Qx* 의 드레인 (drain) 정전용량을 나타내며 그리고 *Cbitwire* 는 비트 선 정전용량을 나타낸다.

- **워드-라인에서의 전력 소비**:

Eword 는 캐쉬 인덱스 (index)의 의한 특별한 한 워드 라인에서 소비되어지는 전력을 나타낸다. 이미 충전된 모든 비트 라인의 캐쉬 태그 부분과 데이타 부분은 인덱스에 의한 하나의 열이 선택되어짐에 따라 원하는 데이타에 대해 읽기 또는 쓰기 동작을 수행할 수 있다. *Eword*에 대한 식은 다음과 같다:

$Eword = V^2 * M * (Nhit + Nmiss) * (8*L + T + C) * (2* Cgate,Q1 + Cwordwire)$, (10)

여기서 *Cwordwire*는 워드선 정전용량 (word-wire capacitance)을 나타내며 식 11로부터 계산되어진다.

$Cwordwire = Ncolumns * (2 * Cgate,Q1 + Cwordwire).$ (11)

- **데이타와 주소 출력 라인에서의 전력 소비**:

*Eoutput*는 외부 버스를 구동시키는 전력 소비를 나타낸다. 즉 캐쉬에서 접근 실패가 발생했을 때 주소나 데이타를 하위 메모리 계층으로 보내거나 또는 데이타를 불러올 때 소비되는 전력을 나타낸다. *Eoutput*를 구하는 식은 크게 두 부분으로 나눌 수 있다:

$$Eoutput = Eaddroutput + Edataoutput, \quad (12)$$

$$Eaddroutput = 0.5 * V^2 * Naddroutput * Caddrout, \quad (13)$$

$$Edataoutput = 0.5 * V^2 * Ndataoutput * Cdaatout, \quad (14)$$

여기서 *Naddroutput* 와 *Ndataoutput*는 각각 주소와 데이타의 출력라인에서의 전체 변이 수를 나타내는 변수이다. 그리고 *Caddrout* 와 *Cdaatout*는 그들에 부합되는 부하 정전용량으로 온-칩에서의 정전용량은 모두 0.5pF, 그리고 오프-칩 상에서의 정전용량은 20pF로 가정하였다 [12]. *Eaddroutput*와 *Edataoutput*는 각각 주소와 데이타 출력에서 소비되는 전력을 나타낸다.

- **주소 입력 라인에서의 전력 소비**:

*Einput*는 주소 디코더의 입력 게이트 상에서 소비되는 전력을 나타낸다. 그러나 이러한 전력 소비는 다른 구성요소에 비해 매우 낮은 전력을 소비함으로 무시할 수 있다.

표3은 이러한 수식을 이용하여 정전용량을 산출한 결과이다. 이러한 결과 값과 제시된 수식을 사용하여 제안한 캐쉬 구조와 victim 캐쉬에 대해서 전력 소비를 비교하고자 한다. 일반적으로 전력 소비에 가장 큰 영향을 미치는 요인은 캐쉬 크기와 접근 실패율이라 할 수 있다. 그러므로 기존의 직접사상 캐쉬와의 비교는 비교적 동일한 캐쉬 크기에 비해 매우 낮은 접근 실패율을 보임으로 이 논문에서 제외시켰다. 또한 분석적 모델을 이용한 정확한 전력 소비 측정을 위해서는 제안된 캐쉬와 victim 캐쉬의 완전 연관 버퍼내의 태그 부분을 CAM으로 계산해야 하지만 두 구조가 동일하기 때문에 계산의 용이성과 비교 분석만을 위해서 집합 연관 캐쉬(set-associative cache)로 가정하였다.

표 3 다양한 정전용량 값

Cdrain,Q1	2.737 fF
Cbitwire	4.4 fF/bitcell
Cdrain,Qp	80.89 fF
Cdrain,Qpa	80.89 fF
Cgate,Q1	0.401fF
Cwordwire	1.8 fF/bitcell
Cg,Qp	38.08 fF
Cg,Qpa, Cg,Qpb	38.08 fF
Caddroutput	0.5pF(on-chip)
Cdataoutput	20 pF(off-chip)

그림 10은 동일한 크기를 가진 제안된 캐쉬와 victim 캐쉬의 전력 소비에 대한 상대적 비교 결과를 보여주고 있다. 이처럼 그림 9, 10에서 알 수 있듯이 제안된 캐쉬는 성능적 측면과 전력 소비적인 측면에서 모두 victim 캐쉬에 비해 다소 우수한 성능을 보임을 알 수 있다. 그림 10에 대한 결과에 대해서 제안된 캐쉬는 8-byte 블록 크기로 동작하는 반면 victim 캐쉬는 32-byte 블록으로 동작하기 때문이며, 또한 메인 캐쉬와 victim 버퍼 사이의 교체 현상이 제안된 캐쉬의 선택적 저장 횟수보다 많이 발생하기 때문이다. 결론적으로 제안된 캐쉬 시스템은 victim 캐쉬에 비해 대략 5%의 전력 소비 감소를 보이고 있다.

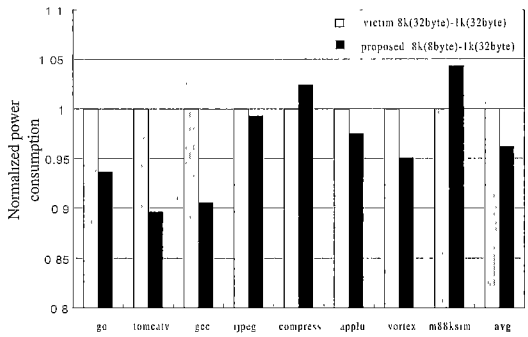

그림 10 제안된 캐쉬와 victim 캐쉬의 전력 소비 상대적 비교

5. 결 론

이 논문의 주목적은 직접사상 캐쉬의 성능 향상을 위한 새로운 캐쉬 시스템을 설계하는 것이며 또한 구현이 간단하고 저 전력과 저 비용에 중점을 두었다. 이러한 연구 목적을 달성하기 위하여 공간적 지역성을 효과적으로 이용할 수 있는 작은 완전 연관 버퍼를 사용하였으며 또한 시간적 지역성의 효과를 극대화시키기 위해 작은 블록 크기와 선택적 저장 방식을 채택하였다. 전력 소비면에서는 큰 블록 크기를 지원하는 직접 사상 캐쉬보다 적은 블록 크기를 지원하는 직접 사상 캐쉬가 읽기 및 쓰기 동작을 할 경우 전력을 적게 소모하며, 또한 쓰기 정정시 예로 한 워드를 위해 32-바이트 큰 블록을 쓰기 정정할 경우 쓰기 이동량 (write-traffic)이 많아져 전력 소모가 큰 단점을 가지게 되나 제안된 캐쉬는 항상 직접 사상 캐쉬에서만 쓰기 정정이 일어남으로 적은 블록 크기인 8-바이트만이 메모리로 쓰여져 쓰기 이동

량을 크게 줄일 수 있는 장점을 가지게 된다.

시뮬레이션 결과에 따르면 제안된 캐쉬의 평균 접근 실패율과 평균 메모리 참조 시간은 기존의 직접사상 캐쉬의 4배보다도 오히려 좋은 성능 향상을 보였으며, 동일한 크기의 victim 캐쉬보다도 나은 성능을 보이고 있다. 또한 전력 소비적인 측면에서도 victim 캐쉬에 비해 대략 5%의 감소 효과를 얻을 수 있었다.

참 고 문 헌

[1] F. Jesus Sanchez, Antonio Gonzalez and Mateo Valero, Static Locality Analysis for Cache Management, *Technical report* UPC-DAC-1997-28.

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

[3] C. Su, and A.Despain, Cache Design tradeoffs for Power and Performance Optimization: A Case study, *ACM/IEEE International Symposium on Low-Power Design*, pp. 63-68, Dana Point, CA, 1996.

[4] J. Kin, M. Gupta, and W. H. Mangione-Smith, The Filter Cache: An Energy Efficient Memory Structure, *MICRO-97: ACM/IEEE International Symposium on Microarchitecture*, pp. 184-193, Research Triangle Park, NC, Dec. 1997.

[5] V. Milutinovic, M. Tomasevic, B. Markovic, M. Tremblay, The Split Temporal/Spatial Cache: Initial Performance Analysis, *SCIzzL-5*, Mar. 1996.

[6] A. Gonzalez, C. Aliagas and M. Mateo, Data Cache with Multiple Caching Strategies Tuned to Different Types of Locality, *Supercomputing '95*, pp. 338-347, July 1995,

[7] G. Kurpanchek et al, PA-7200: A PA-RISC Processor with Integrated High Performance MP Bus Interface, *COMPCON Digest of Papers*, pp. 375-382, Feb. 1994.

[8] James E. Bennett, and Michael J. Flynn, Prediction Caches for Superscalar Processors, *Proceedings of the 30th Annual Int'l Symposium on Micro-architecture*, pp. 81-90, Nov. 1997.

[9] Jude A. Rivers, and Edward S. Davidson, Reducing Conflicts in Direct-Mapped Caches with a Temporality-Based Design, *Proceedings of the 1996 International Conference on Parallel Processing*, Vol. I, pp. 151-162, Aug. 1996.

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

[11] Johannes M. Mulder, N.T. Quach, and Michel J. Flynn, An Area Model for On-Chip Memories and its Applications, *IEEE journal of solid state Circuits*, 26(2) pp. 98-106, Feb. 1991.

[12] S. J. E. Wilton, and N. Jouppi, An Enhanced Access and Cycle Time Model for On-Chip Caches, *Digital WRL Research Report 93/5*, July 1994.

[13] M. B. Kamble, and K. Ghose, Energy-Efficiency of VLSI Cache: A Comparative Study, *in Proc. IEEE 10-th. Intl. Conf. On VLSI Design*, pp. 261-267, Jan. 1997.

[14] M. B. Kamble, and K. Ghose, Analytical Energy Dissipation Models for Low Power Caches, *ACM/IEEE International symposium on Low-Power Electronics and Design*, Aug. 1997.

이 정 훈

1999년 성균관대학교 제어계측공학과(학사). 1999 ~ 현재 연세대학교 컴퓨터과학과(석사과정). 관심분야는 지능형 메모리 시스템, 신경망 네트워크, 고성능 컴퓨터 구조임.

이 장 수

1994년 한양대학교 전자계산학과(학사). 1998년 연세대학교 컴퓨터과학과(석사). 1998년 ~ 현재 연세대학교 컴퓨터과학과(박사과정). 관심분야는 고성능 컴퓨터 구조, 프로세서-메모리 집적구조, 성능평가임.

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 부교수. 관심분야는 병렬처리 시스템, 컴퓨터구조, Heterogeneous computing임.