고성능 내장형 프로세서의 에너지 소비 감소를 위한 데이타 캐쉬 통합 설계 방법

(Hybrid Scheme of Data Cache Design for Reducing Energy Consumption in High Performance Embedded Processor)

심 성 훈 † 김 철 홍 † 장 성 태 †† 전 주 식 †††
(Sunghoon Shim) (Cheol Hong Kim) (Seong Tae Jhang) (Chu Shik Jhon)

요 약 현재 내장형 프로세서에서 캐쉬 사이즈는 더 많은 트랜지스터 집적도와 낮은 공급 전력에 기인하여 점점 더 증가 되어지는 추세이다. 하지만 캐쉬 사이즈가 커질수록 더욱 더 많은 에너지 소비가 발생하게 되며, 결과적으로 프로세서 전체에서 소비하는 에너지 중에서 캐쉬에서 소비되는 에너지의 비중이 점점 더 증가 되고 있다. 이에 따라 캐쉬 에너지 소비를 줄이기 위한 많은 기법들이 제시되어져 왔다. 하지만 이러한 기존의 기법들은 캐쉬 에너지 소비의 2가지 방면, 즉, 정적 캐쉬 에너지 소비와 동적 캐쉬 에너지 소비 중에서 어느 한쪽에 초점을 맞추어 제시되어진 기법들이었다. 본 논문에서는 고성능 내장형 프로세서에서 캐쉬 에너지 소비의 2가지 방면인, 정적 캐쉬 에너지 소비와 동적 캐쉬 에너지 소비를 동시에 감소시키는 정적 에너지 소비 감소와 동적 에너지 소비 감소의 통합 기법을 제안한다. 이 통합 기법에는 이미 제안되어진 두 가지 기법, 동적 에너지 소비를 감소시키기 위한 웨이 예측 기법과 정적 에너지 소비를 감소시키기 위한 드라우지 캐쉬(drowsy cache) 기법을 적용한다. 또한 드라우지 캐쉬 기법을 사용하였을 때 생기는 추가적인 프로그램 실행 사이클들을 줄이기 위한 "프로그램 카운트를 이용하는 드라우지 상태의 데이타 캐쉬 라인 미리 깨움" 기법을 제안한다. 이러한 기법 적용을 레벨 1 데이타 캐쉬에 적용한다. 제안 되어진 통합 기법을 통해서 정적 데이타 캐쉬 에너지 소비와 동적 데이타 캐쉬 에너지 소비를 동시에 줄일 수 있게 되며, 같이 제안되어진 "드라우지 상태의 데이타 캐쉬 라인 미리 깨움"기법은 통합 기법 때문에 발생하는 추가적인 프로그램 실행 사이클의 증가를 감소시킬 수 있다.

키워드 : 데이타 캐쉬, 정적 캐쉬 에너지, 동적 캐쉬 에너지, 드라우지 캐쉬, 웨이 예측

Abstract The cache size tends to grow in the embedded processor as technology scales to smaller transistors and lower supply voltages. However, larger cache size demands more energy. Accordingly, the ratio of the cache energy consumption to the total processor energy is growing. Many cache energy schemes have been proposed for reducing the cache energy consumption. However, these previous schemes are concerned with one side for reducing the cache energy consumption, *dynamic cache energy only*, or *static cache energy only*. In this paper, we propose a hybrid scheme for reducing dynamic and static cache energy, simultaneously. for this hybrid scheme, we adopt two existing techniques to reduce static cache energy consumption, drowsy cache technique, and to reduce dynamic cache energy consumption, way-prediction technique. Additionally, we propose a early wake-up technique based on program counter to reduce penalty caused by applying drowsy cache technique. We focus on level 1 data cache. The hybrid scheme can reduce static and dynamic cache energy consumption simultaneously, furthermore our early wake-up scheme can reduce extra program execution cycles caused by applying the hybrid scheme.

Key words : data cache, static cache energy, dynamic cache energy, drowsy cache, way-prediction

† 학생회원 : 서울대학교 전기컴퓨터공학부
shshim@panda.snu.ac.kr
: kimch@panda.snu.ac.kr
†† 종신회원 수원대학교 컴퓨터학과 교수
stjhang@suwon.ac.kr
††† 종신회원 : 서울대학교 전기컴퓨터공학부 교수
csjhon@panda.snu.ac.kr
논문접수 : 2004년 9월 24일
심사완료 : 2005년 11월 25일

1. 서 론

현재, 많은 내장형 시스템들이 우리 주변에서 사용되고 있다. 특히 높은 계산 능력을 요구하는 내장형 시스템들의 사용이 폭발적으로 증가하고 있는 추세이다. 이러한 고성능 내장형 시스템의 사용의 증가는 폭발적인 모바일기기의 증가와 핸드-핼드 PC 그리고 네트워크 시스템의 증가에 기인한다. 이러한 내장형 시스템에서는 전력 소비를 줄이는 것이 매우 중요한 이슈이며, 따라서 사용되는 프로세서에서의 파워 소비를 감소시키는 것은 내장형 시스템을 위한 프로세서 디자인의 주된 이슈중의 하나가 되고 있다. 또, 기술의 발달과 고성능을 요구하는 내장형 시스템으로 인하여 프로세서 내부의 캐쉬 크기가 점점 증가하고 있는 실정이다. 하지만 이러한 캐쉬 크기의 증가는 그 만큼의 에너지 소비의 증가를 가져오게 된다.

이러한 캐쉬 에너지 소비를 줄이기 위해 많은 기법들이 소개되어 왔다. 웨이 예측 기법도 그 중의 하나이다[1]. 웨이 선택 기법은 예측 테이블을 사용하여 접근 되어질 캐쉬의 웨이를 미리 예측하여 해당 되는 웨이에만 에너지를 공급하는 기법이다. 웨이 예측 기법에는 명령어 프로그램 카운터(PC)를 이용하여 예측 테이블을 접근하는 방식과 로드 소스 레지스터와 명령어의 옵셋을 배타적 OR(XOR) 시킨 결과 값으로 예측 테이블을 접근하는 방식이 있다[2]. 이 방식들은 모두 캐쉬의 동적 에너지를 줄이기 위한 방법들이다.

위의 동적 캐쉬 에너지를 줄이는 기법 외에 정적 캐쉬 에너지를 줄이기 위한 기법들이 제안되어졌다. 몇 몇 기법은 공급 전력(supply voltage)을 줄이는 방식을 취하고 있다[3,4]. 이러한 공급 전력을 줄여 정적 캐쉬 에너지를 감소시키는 방식 중, 드라우지(drowsy) 캐쉬 기법은 다단계의 공급 전력을 이용하여 캐쉬의 정적 에너지를 줄이고 있다[5]. 이 기법은 각각의 캐쉬 라인을 저전압이 공급되는 상태인 드라우지(drowsy) 상태와 일반적인 전압이 들어가는 상태인 일반 상태의 두 가지의 상태중의 하나를 갖게 하여 캐쉬의 정적 에너지 사용을 줄이는 방법이다. 이 캐쉬 기법에서 만약 저전압 상태인 드라우지(drowsy) 상태의 캐쉬 라인안의 데이타를 요구하게 되면, drowsy 상태에서 일반상태로 돌아온 후에 데이타를 제공하게 된다. 이러한 과정을 깨움(wakeup)이라고 한다. 이렇게 낮은 전압을 공급하는 상태인 드라우지(drowsy)상태로 캐쉬 라인들을 일정 프로세서 사이클마다 전이시켜서 전체적인 캐쉬 정적에너지의 사용을 줄이게 된다.

지금까지의 캐쉬 에너지 사용을 줄이기 위한 여러 기법들을 살펴보면, 각 기법들이 동적 캐쉬 에너지 소비만을 감소시키거나 또는 정적 캐쉬 에너지 소비만을 감소시키는 방식을 사용해왔다. 따라서 본 논문에서는 동적 캐쉬 에너지 소비와 정적 캐쉬 에너지 소비를 동시에 감소시키는 통합 방법을 제시한다. 이 통합 방법에는 이미 제시되어 있던 2가지의 캐쉬 에너지 소비 감소 기법인, 드라우지 캐쉬(drowsy cache) 기법과 프로그램 카운터(Program Counter)를 이용하는 캐쉬 웨이(way) 예측 기법을 적용하였다. 제안 되어진 통합 기법에서 캐쉬 웨이(way) 예측 기법은 동적 캐쉬 에너지 소비를 감소시키며, 동시에 드라우지 캐쉬(drowsy cache) 기법은 정적 캐쉬 에너지의 소비를 감소시킨다. 이러한 제시되어지는 통합 방법은 레벨 1 데이타 캐쉬에 적용하였다. 하지만 앞에서 제안되어진 통합 방법은 캐쉬 웨이 기법에서 웨이 예측 실패로 인한 추가적인 프로그램 실행 사이클과 드라우지 캐쉬 적용에서 캐쉬 라인 깨움에 따른 추가적인 프로그램 실행 사이클의 증가를 야기하게 된다. 따라서 이러한 추가 적인 프로그램 실행 사이클의 증가를 감소시키기 위한 기법도 함께 제시한다. 이 기법은 캐쉬 라인의 데이타가 요구되어지는 시점보다 일찍 드라우지 상태의 캐쉬 라인을 깨움(wakeup)으로써 정적 캐쉬 에너지 감소를 위해 적용한 드라우지 캐쉬 사용 시에 추가적으로 드는 사이클을 줄일 수 있다. 이 논문의 구성은 다음과 같다. 2장에서는 이미 제시되어진 관련된 연구에 대해서 언급하며, 3장에서는 이 논문에서 제시하는 기법에 대한 모델링과 방법론에 대해서, 4장에서는 그 결과에 대해서 논하며, 마지막으로 5장에서 결론에 대해서 논한다.

2. 캐쉬 에너지 사용에 대한 감소 기법들

현재 사용되어지는 많은 프로세서들은 충돌 캐쉬 미스(conflict cache miss)를 줄이기 위하여 직접 사상 캐쉬(direct mapped cache)보다는 셑-어소세이티브 캐쉬를 사용한다[6]. 셑-어소시에이티브 캐쉬에서 하나의 셑은 그 캐쉬의 웨이 수와 같은 수의 캐쉬 블록들을 가진다. 셑-어소시에이티브 캐쉬는 프로세서 코어에서 데이타가 요청되었을 때, 해당 셑의 모든 캐쉬 블록들을 위해 동적 에너지가 사용된다. 하지만 캐쉬 블록들의 태그 비교가 끝난 후에는 웨이 수에 해당하는 블록 중에서 하나의 블록만이 데이타를 요구한쪽에 전달되어지게 되며, 그 전달되어지는 하나의 블록을 제외한 나머지 블록들은 무시하게 된다. 결과적으로 셑-어소시에이티브 캐쉬가 직접 사상 캐쉬(direct mapped cache)보다 충돌 미스는 줄일 수 있지만, 더욱 많은 동적 캐쉬 에너지를 소비하게 된다.

2.1 동적 캐쉬 에너지 소비의 감소

앞선 셑-어소시에이티브 캐쉬 사용시 에너지의 소비를 감소시키기 위한 연구에는 크게 2가지가 있다. 동적

에너지 소비 감소에 관한 부분과 정적 에너지 소비 감소에 관한 부분이다. 동적 캐쉬 에너지 소비는 웨이 예측 기법으로 감소시킬 수 있다. 이 기법에서는 셀-어소시에이티브 캐쉬에서 같은 셀 안의 여러 블록 중 예측한 하나의 블록만 접근하는 기법이다. 따라서 같은 셀의 나머지 블록 접근에 대한 동적 에너지 소비를 없앨 수 있게 된다. 태그 체크 후에 만약 그 예측이 성공하였다면 해당 캐쉬에 대한 접근은 성공한 것이 되고, 예측이 실패하였다면, 해당 캐쉬 셀 중에서 예측한 블록을 제외한 나머지 캐쉬 블록들에 대해 접근이 일어나게 된다. 이에 따라 여분의 1 프로세서 사이클을 필요로 한다.

동적 캐쉬 에너지의 사용을 감소시키기 위한 웨이 예측 기법으로는 두 가지가 있다. 하나는 명령어 프로그램 카운터(Program Counter)로 예측 테이블을 탐색하는 방법이다. 이 기법에서는 명령어 프로그램 카운터가 테이블에서 하나의 엔트리를 찾기 위한 인덱스로 사용되어지며, 그 엔트리는 예측하는 웨이 넘버가 들어 있다. 다른 기법은 로드 소스 레지스터 값과 오프셋을 배타적 오어링(xoring)시켜서 나온 값으로 예측 테이블을 접근하는 방법이다. 그림 1에서 위의 두 가지 방법에 대한 타이밍도를 보이고 있다. 명령어 프로그램 카운터를 이용하는 방법은 프로그램 카운터로 예측 테이블을 접근하기 때문에 한 명령어가 같은 캐쉬 블록을 반복적으로 접근할 때에 잇점을 가진다. 또한 이 기법은 접근되어질 캐쉬 웨이에 대한 정보를 실제 캐쉬 접근 시점인 파이프라인의 MEM단계보다 빨리 알 수 있기 때문에 타이밍상의 잇점이 있다. 다른 방식인 로드 소스 레지스터 값과 오프셋을 배타적 오어링(xoring) 시키는 방법은 예측의 적중 확률이 프로그램 카운터방식보다는 높다는 장점이 있다. 하지만 타이밍상, 실제 캐쉬 접근 시점 전에 접근되어질 캐쉬에 대한 정보(예측 웨이)를 알아내기가 힘들다는 단점이 있다. 앞의 두 기법 모두 예측 실패

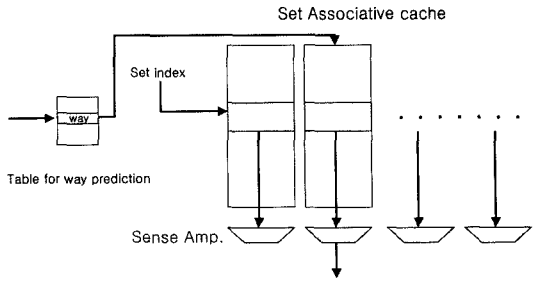

그림 2 기본 웨이 예측 기법의 구조

시에는 1 프로세서 사이클이 더 요구된다. 그림 2에서는 두 가지 웨이 예측 기법으로 웨이를 예측한 이후의 과정을 보이고 있다. 그림에서 보이는 바와 같이 접근이 예측되는 웨이에 대한 정보를 얻은 후, 그 해당 웨이에 대해서만 캐쉬 접근이 일어나게 된다.

2.2 정적 캐쉬 에너지 소비 감소

정적 캐쉬 에너지의 소비를 감소시키기 위한 여러 가지 기법들이 소개되어져 왔다[3-5]. Cache Decay 방식도 그러한 방식들 중의 하나이다[4]. Cache Decay 방식은 자주 사용되지 않는 캐쉬 라인의 전력 공급을 정기적으로 중단하는 방식을 사용한다. 공급 전력이 중단되어진 캐쉬 라인들안의 데이타는 해당 캐쉬 라인의 전력 공급이 중단되어져 있기 때문에 재사용 되어질 수 없다. 만약 전력 공급이 중단된 캐쉬 라인에 대한 접근이 발생한다면, 해당 캐쉬 라인에 전력이 재공급되면서 다음 단계의 캐쉬 혹은 메모리에서 해당 데이타를 가지고 와야만 한다. 드라우지 캐쉬(drowsy cache) 기법도 정적 캐쉬 에너지 소비를 줄이기 위한 기법들 중의 하나이다[5]. 이 기법은 캐쉬 라인의 전력 공급을 중단하는 대신에 정기적으로 모든 캐쉬 라인의 상태를 일반적 공급전력보다 저전력을 공급하는 상태로 두고 해당 캐쉬 라인에 대한 접근이 발생할 때에만 저전력 상태에서 일반적

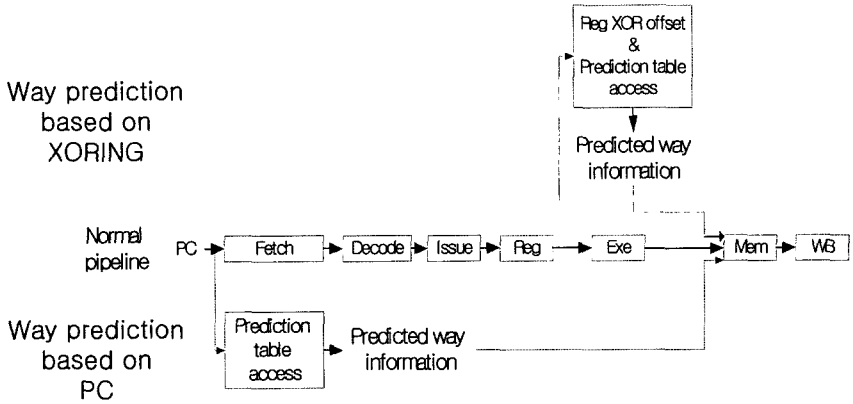

그림 1 기본 웨이 예측 기법의 타이밍도

인 전력 공급 상태로 돌아오게 하는 기법이다. 여기서 저전력을 공급하는 캐쉬 라인 상태를 드라우지 상태(drowsy mode)라고 한다. 앞서 설명한 Cache Decay 기법과 달리 드라우지 캐쉬는 드라우지 상태를 갖는 캐쉬 라인 안의 데이타 값들을 유지한다. 따라서 드라우지 상태의 캐쉬 라인에 대한 접근이 발생할 시에는 다음 단계의 캐쉬나 메모리에서 데이타를 가져오는 대신, 저전력 공급 상태인 드라우지 상태에서 평상시의 상태(평상시의 전력을 공급하는)로 상태를 바꾸어 해당 캐쉬 라인의 데이타를 읽어오게 된다. 그림 3은 이러한 드라우지 캐쉬의 구조를 보이고 있다. 크게 각 캐쉬 라인당 한 비트의 드라우지 비트와 공급전력 제어부로 나뉘며, 일정 프로세서 사이클마다 각 캐쉬 라인은 드라우지 비트의 설정으로 드라우지 상태로 전이가 되며, 캐쉬 라인에 대한 접근이 발생한다면, 드라우지 비트의 값을 확인하여 드라우지 상태인지를 확인하여, 해당 드라우지 비트값이 저전력 상태인 드라우지 상태라면 공급 전력 제어부에서 일반적인 전력 공급을 하게 하여 드라우지 상태 캐쉬 라인을 깨우는 구조로 되어 있다. 드라우지 캐쉬 상태(drowsy cache mode)를 일반적인 캐쉬 상태(normal cache mode)로 바꾸는 과정을 "깨움(wake-up)" 과정이라고 한다. 하지만 이러한 깨움 과정은 추가적인 1 프로세서 사이클을 필요로 한다. 따라서 드라우지 캐쉬를 적용시에는 일반 캐쉬 적용보다 프로그램 실행 사이클이 증가하게 된다.

3. 모델링과 방법론

3.1 정적 에너지 소비과 동적 에너지 소비의 동시 감소

앞서 2.1절과 2.2절에서 캐쉬의 에너지 소비를 줄이기 위한 기존 기법들이 소개되었다. 기존기법 중 웨이 예측(way prediction)기법은 동적 캐쉬 에너지 소비를 줄이기 위한 기법이었고 드라우지 캐쉬 (drowsy cache) 기법은 정적 캐쉬 에너지 소비를 줄이기 위한 기법이었다. 하지만 이미 제안되어진 많은 기법들은 단지 캐쉬 에너지 소비의 한 방면, 즉, 동적 캐쉬 에너지 소비만 감소시키거나 정적 캐쉬 에너지 소비만 감소시키는 데에만 주안점을 두고 제안되어졌다. 이는 동적 캐쉬 에너지 소비를 줄이기 위해서 기존의 웨이 예측 기법을 적용하게 될 때에는 예측 실패로 인한 전체적인 프로그램 실행 사이클을 증가시켜서, 동적 캐쉬 에너지 소비를 줄이는 것과 동시에 정적 캐쉬 에너지 소비를 증가 시키게 되고, 또한 정적 캐쉬 에너지 소비를 줄이기 위한 드라우지 캐쉬(drowsy cache)방법도 하나의 드라우지 상태의 캐쉬 라인들을 깨우는데 추가적인 1 프로세서 사이클을 필요로 하고, 이 추가적인 사이클들로 인해 프로세서 파이프라인 단계에서 데이타를 기다리던 명령어들을 다시 이슈시키거나, 아예 다시 처음부터 실행시켜야 되는 상황이 발생하게 되어 동적 전력 소비의 증가를 가져올 수 있다[7]. 때문에 본 논문에서 제안되어지는 저전력 캐쉬를 위한 통합 방식은 기존에 이미 제안되어진 방식인, 동적 캐쉬 에너지 소비를 줄이기 위한 웨이 예측 방식과 정적 캐쉬 에너지 소비를 줄이기 위한 드라우지 캐쉬방식을 결합시켜 동적 캐쉬 에너지와 정적 캐쉬 에너지 소비를 동시에 줄이기 위한 방식이다. 이렇게 기존 두 가지의 방식을 같이 적용하였을 때에는 두 방식의 정적 캐쉬 에너지 소비 감소와 동적 캐쉬 에너지 소비 감소를 동시에 할 수 있지만, 반대로 두 방식 때문에 발생하는 성능 감소도 동시에 가져오게 된다. 따라서 제안되어진 통합방식에서 발생하는 추가적 프로그램 사이클 증가에 따른 성능감소를 줄이기 위하여 기존 드라우지

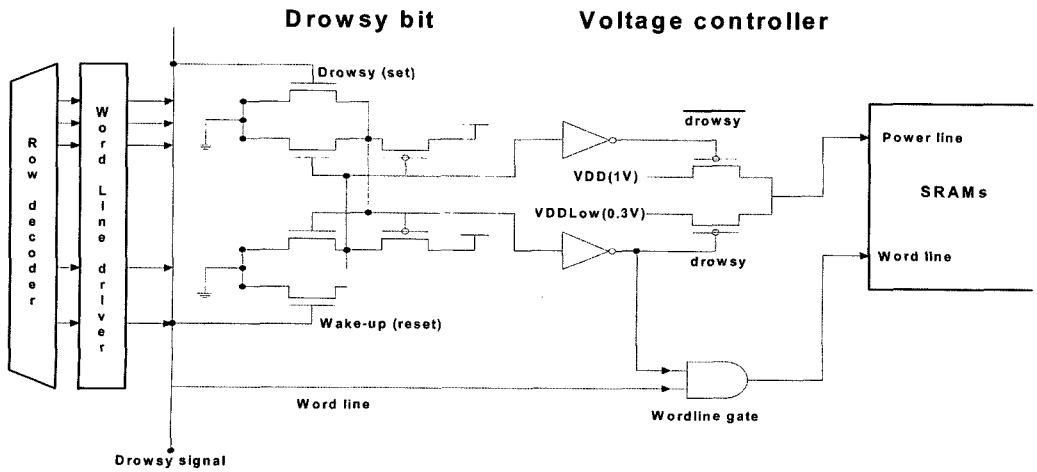

그림 3 드라우지 캐쉬의 구조도

캐쉬 방식을 개선시킨 방식을 제안하여 적용하였다. 이 개선된 드라우지 캐쉬 방식은 기존의 드라우지 캐쉬 방식에서 하나의 드라우지 상태에 있는 캐쉬 라인을 깨우는 데 필요로 되어지는 추가적인 1 프로세서 사이클을 줄이기 위한 기법이다.

제안되어진 통합 방식에서 동적 캐쉬 에너지 소비를 줄이기 위한 방식으로는 기존에 제안되어졌던 웨이 예측 방식을 적용하였으며, 웨이 예측 방식 중 명령어 프로그램 카운터(Program Counter)를 사용하는 방식을 적용하였다. 이 방식은 2.1절에서 설명한바와 같이 한번 접근했던 데이타 캐쉬의 웨이를 저장해 둔 테이블을 이용한다.

그림 4는 제안되어진 통합방식에서 사용되어지는 예측 테이블의 엔트리 구조를 보이고 있다. 예측 테이블의 각 엔트리는 2개의 필드를 가지고 있다. 먼저 set index 필드는 제안되어진 드라우지 캐쉬 상태의 라인을 미리 깨우기 위한 해당 캐쉬 라인의 인덱스 저장 필드이다. way-number 필드는 웨이 예측을 위한 예측되어진 웨이값을 위한 필드이다.

Set Index	Way-number

그림 4 통합 방식에서 사용되는 예측 테이블의 엔트리 구조

예측 테이블의 각 엔트리의 값은 초기에는 0를 갖게 되며, 한 캐쉬 라인이 요청되고 그 캐쉬 라인의 접근이 성공하게 되면, 해당 캐쉬의 웨이 정보와 셀의 인덱스 값으로 예측 테이블 엔트리의 내용을 갱신하게 된다.

그림 5와 6에서 하나의 드라우지 캐쉬 상태의 라인을 미리 깨우는 과정과 웨이 예측과정을 보이고 있다.

레벨 1 데이타 캐쉬 라인이 2개의 웨이를 가졌다고 가정했을 때, 먼저 명령어의 프로그램 카운터(PC)값으로 예측 테이블의 한 엔트리를 접근하게 된다. 이 엔트리는 위에서 설명한 바와 같이 2개의 필드를 가지고 있다. 즉, 드라우지 상태에 있는 한 라인을 접근하기 위한 set index와 접근할 웨이를 예측한 값인 웨이 넘버가 바로 그것이다. 예측 테이블의 엔트리에 접근해서 이 2가지 값을 얻게 된 후, 웨이 예측 값으로 파이프 라인의 캐쉬 접근 스테이지에서 예측되어진 하나의 웨이만을 접근하게 되며, 이는 그림 5의 캐쉬 데이타 어레이중 우측 부분의 음영이 있는 곳에 해당된다. 또한 set index 값으로는 하나의 드라우지 캐쉬 상태에 있는 캐쉬 라인을 그림 6의 "mem" 스테이지보다 더 빠른 스테이지(예를 들어 "EXE" 스테이지)에서 미리 깨우게 된다. 이렇게 미리 깨우게 되면 기본적인 드라우지 캐쉬 기법에서 깨울 때 마다 필요로 되어지는 1 사이클의 패널티를 줄일 수 있게 된다. 결국 전체적인 프로그램 실행 사이클이 원래의 드라우지 캐쉬 기법만 적용 하였을 때보다

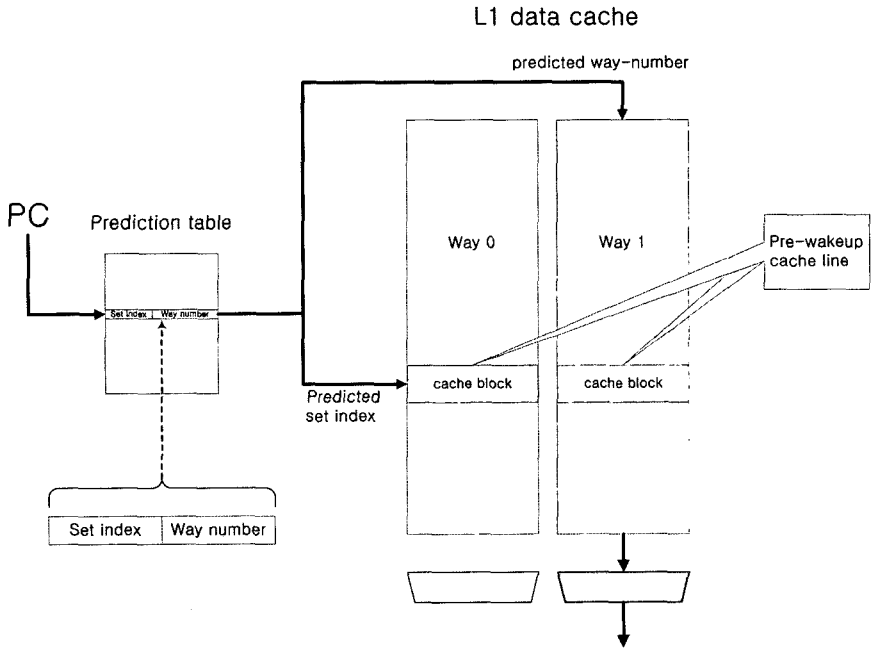

그림 5 통합 방식에서의 웨이 예측 과정과 드라우지 상태 캐쉬 라인의 미리 깨움 과정

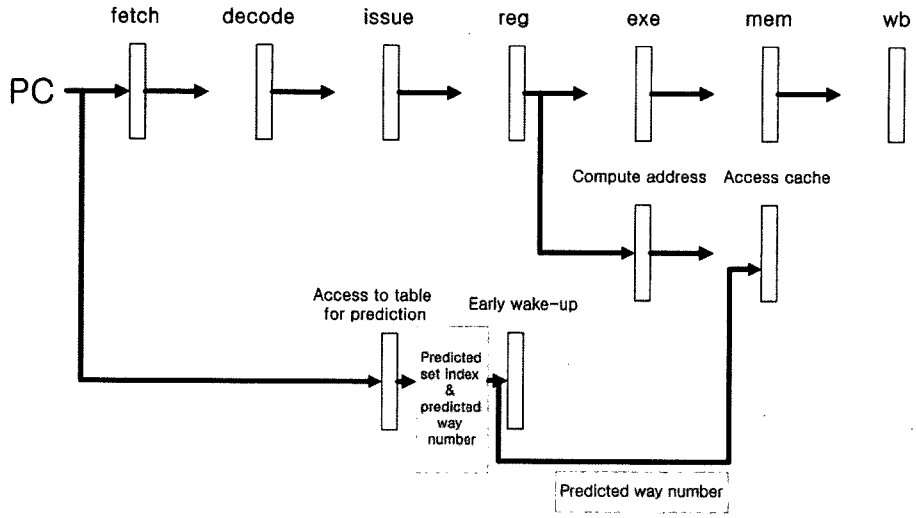

그림 6 통합 방식의 웨이 예측과 드라우지 상태의 캐쉬 미리 깨움의 타이밍도

줄어들게 되어 그 만큼의 캐쉬 에너지 소비도 줄일 수 있게 된다. 또한 웨이 예측 기법을 사용으로 인해 실제 캐쉬 접근시, 여러개(위의 예에서는 2개)의 웨이중 하나의 웨이만을 접근하게 됨으로써 동적 캐쉬 에너지 소비도 줄일 수 있게 된다.

제안되어진 통합 방식에서도 기본적인 웨이 기법에서처럼 웨이 예측의 실패시, 1 프로세서 사이클이 추가적으로 필요로 하게 되며, 드라우지 캐쉬 라인을 미리 깨울 때에도, 예측이 틀려서 다른 라인을 미리 깨우게 되었을 때에는 일반적인 드라우지 캐쉬와 같이 동작하여, 일반적인 드라우지 캐쉬 기법에서와 마찬가지로 캐쉬를 깨우기 위한 추가적인 1 사이클이 더 걸리게 된다.

3.2 실험 환경

실험을 위해 Wattch 시뮬레이터를 사용하였다[8]. 이 Wattch 시뮬레이터는 SimpleScalar를 바탕으로 작성되어진 모의 실험도구이다[9]. 하지만 Wattch 시뮬레이터는 단지 프로세서의 동적 에너지 소비에 대한 결과값만을 제공하기 때문에 제안되어진 기법을 위해 Wattch 시뮬레이터를 수정하였다. 모델링된 프로세스는 사이클당 2 이슈를 하며, 32KB, 1 사이클, 2 웨이의 레벨 1 명령어 캐쉬를 가지며, 같은 사이즈인 32KB, 1 사이클, 2 웨이의 레벨 1 데이타 캐쉬를 갖는다. 또한 256KB, 8 사이클, 4웨이의 통합 레벨 2 캐쉬를 갖는다. 표 1에서 자세한 모의 실험 인자에 대해 보이고 있다.

실험에 사용된 벤치마크 프로그램으로는 SPEC2000이 사용되었다[10]. 표 2에서 보여 지는 것처럼 SPEC2000의 벤치 마크 프로그램중 SpecINT2000에서 10개를, SpecFP2000에서 10개를 선택하여 사용하였으며, 20개의 벤치 마크 프로그램에 대해 입력 셀은 test 입력셀으로 하여 각 프로그램의 수행 종료까지 실행시켜 결과값을 도출하였다. 에너지에 대한 실험 결과값들은 모두 Wattch 시뮬레이터가 제공하는 값을 사용하였다.

표 1 모의 실험 인자

Parameter	value
CPU	2 issues per cycle
L1 I-cache	32KB, 2-way, 32 byte block, 1 cycle latency
L1 D-cache	32KB, 2-way, 32 byte block, 1 cycle latency
L2 cache	Unified cache, 4-way, 256KB, 64 byte blocks, 8 cycle latency
Memory	64 cycle latency
Instruction TLB	16-entry
Data TLB	32-entry
Prediction table for way and wake-up	1024, 512, 256, 128, 64 entry

표 2 벤치마크 프로그램

SPEC2000			
SpecINT2000	gcc	SpecFP2000	ammp
	gzip		art
	bzip2		equake
	mcf		wupwise
	parser		swim
	vortex		mgrid
	vpr		applu
	crafty		galgel
	eon		lucas
	gap		apsi

본 실험에서는 제안되어진 저전력 캐쉬를 위한 통합

방식의 시스템과 기본적인 캐쉬를 사용하는 시스템, 그리고 드라우지 캐쉬 라인의 미리 깨움과 비교하기 위한 기본적인 드라우지 캐쉬 시스템을 모델링하여 모의 실험을 수행하였다. 여기서 말하는 기본적인 캐쉬 모델이란 어떠한 캐시 에너지 소비 감소 기법도 사용되어지지 않은 순수한 캐쉬 모델을 의미한다. 본 논문에서 제안되어진 저전력 캐쉬를 위한 통합 방식 모델에서 사용되어진 드라우지 캐쉬 기법은 "awaking tags" 방식이며, 이는 레벨 1 데이타 캐쉬에서 태그를 제외하고 오직 데이타 어레이에만 드라이지 캐쉬 기법을 적용한 것을 의미한다. 또한 웨이 예측과 드라우지 상태 캐쉬 라인의 미리 깨움을 위해서 각각 1024, 512, 256, 128, 64 엔트리 개수를 갖는 예측 테이블을 사용하였다. 제안되어진 저전력 캐쉬를 위한 통합 기법 모델과 기본적인 드라우지 캐쉬 모델에서 모든 캐쉬 라인은 2000 사이클마다 드라우지 상태로 전이하게 된다.

4. 실험 결과 및 분석

4.1 드라우지 캐쉬의 라인 미리 깨움

2.2절에서 언급한대로, 드라우지 캐쉬는 드라우지 상태라 불리우는 특별한 저전력을 소비하는 캐쉬 상태를 이용하여 정적 캐쉬 에너지 소비를 줄인다. 드라우지 캐쉬에서는 주기적으로 모든 캐쉬 라인을 저전력 상태인 드라우지 상태로 바꾸게 된다. 만약 드라우지 상태의 캐쉬에 대한 접근 요구가 발생한다면, 그 요구 되어지는 캐쉬 블록을 포함하는 캐쉬 라인을 깨운 후(wake-up), 접근을 하게 된다. 하지만 기본적으로 드라우지 캐쉬에서 이러한 하나의 드라우지 상태의 캐쉬 라인을 깨우는 과정에는 추가적인 1 사이클을 필요로 하게 되고, 이러한 추가적인 사이클들이 누적되어서 결과적으로 전체적인 프로그램 실행 시간을 증가시키게 됨으로서 시스템의 성능 하락을 가져오게 된다. 3장에서 이미 설명된 대로, 본 논문에서는 이러한 추가적인 사이클들을 줄이기 위해 "프로그램 카운트를 이용하는 드라우지 상태의 캐쉬 라인 미리 깨움" 기법을 제안하였다. 이 기법은 드라우지 상태의 캐쉬 라인 접근시 실제 파이프 라인의 MEM 단계보다 이전 단계에서 깨움으로써, 즉 다른 파이프라인의 단계와 깨움을 중첩시킴으로써 드라우지 상태의 캐쉬 라인에 대한 접근 횟수를 줄이게 되는 방식이며, 이것은 곧 드라우지 상태의 캐쉬 라인들을 깨우는데 필요로 되는 추가적인 사이클을 없앨 수 있게 된다는 것을 의미한다.

이번 절에서는 제안되어진 "미리 깨움" 기법에 의해 얼마만큼의 비율로 드라우지 상태의 캐쉬라인 접근이 줄어드는 지를 보이게 된다. 제안되어진 "미리 깨움" 기법은 요구되어진 드라우지 상태의 캐쉬라인을 실제 캐쉬 라인 접근 단계보다 이전에 깨우게 되며, 여기에는 어떠한 성능 감소도 없다. 단, 만약 미리 깨운 캐쉬 라인이 요구되어진 캐쉬 라인이 아니라면, 이 후의 과정은 기본 드라우지 캐쉬에서처럼 추가적인 1 사이클을 필요로 하며, 잘못 깨운 캐쉬 라인으로 인한 약간의 정적 에너지 손실은 발생될 수 있다. 이에 대해서는 4.2절에서 보인다.

그림 7에서 각 벤치 마크 프로그램에서 가장 좌측의 바(bar)가 일반적인 드라우지 캐쉬에서의 드라우지 상태 캐쉬 라인에 대한 접근을 100으로 본 바(bar)이며, 오른쪽 바(bar)들은 각각 1024, 512, 256, 128, 64 엔트리 개수의 예측 테이블을 이용하였을 때의 결과를 보이고 있다. 제안 되어진 "미리 깨움"을 적용 하였을 때, 평균적으로 64엔트리의 예측 테이블에서는 20.4%, 128엔트리에서는 25.5%, 256엔트리에서는 30.6%, 512엔트리에서는 34.5%, 그리고 1024엔트리에서는 37.1%의 "드라우지

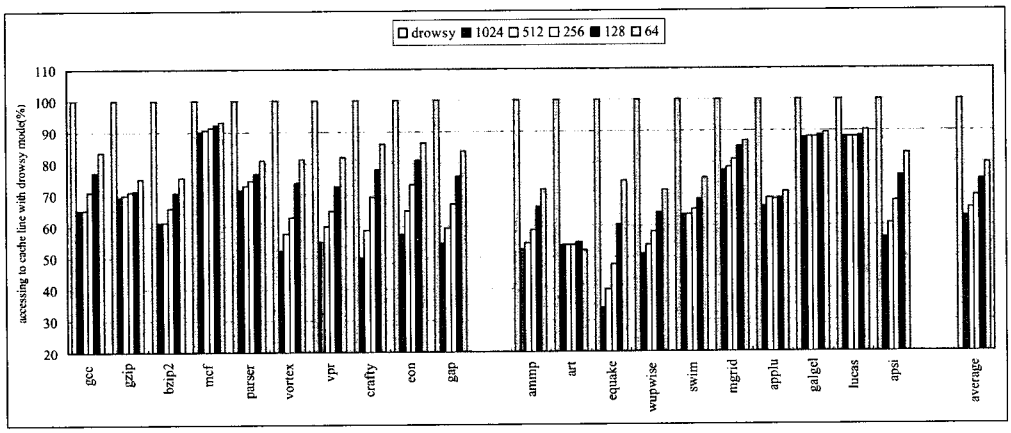

그림 7 드라우지 상태의 캐쉬라인들의 접근 횟수에 대한 비율

상태의 캐쉬 라인 접근 횟수"의 감소를 보이고 있다. 이 감소 비율은 기본 드라우지 캐쉬에서 드라우지 상태의 캐쉬 라인 접근 횟수와 비교된 비율을 뜻한다. 결국 위의 각 테이블 사이즈에서의 드라우지 상태 캐쉬 라인 접근 횟수에 대한 감소 비율은 그 만큼의 추가적인 사이클을 감소시키게 됨을 의미한다.

그림 7의 대부분의 벤치 마크 프로그램에서 큰 엔트리 개수를 갖는 예측 테이블을 이용하였을 때, 상대적으로 높은 비율로 드라우지 상태의 캐쉬 라인 접근 횟수가 줄어드는 것을 알 수 있다. 다만 galgel, lucas, gzip, 그리고 art같은 SpecFP2000의 벤치마크 프로그램들에서는 예측 테이블의 크기가 증가하여도 그 만큼의 드라우지 상태의 캐쉬 라인 접근 횟수에 대한 비율이 증가하고 있지 않은데, 이것은 하나의 명령어 프로그램 카운터가 같은 캐쉬 블록을 반복적으로 접근하지 않는 벤치마크 프로그램들의 캐쉬 접근 패턴에 기인한다.

4.2 정적 캐쉬 에너지 소비의 감소

이번 절에서는 제안 되어진 통합 방식이 얼마나 많은 정적 캐쉬 에너지 소비를 줄일 수 있는 지를 보이게 된다. 이러한 정적 캐쉬 에너지 소비 감소에 대한 척도로 "전체 캐쉬 라인중 드라우지 상태의 캐쉬 라인의 비율"이라는 것을 제안한다. 이 척도는 일정 프로세서 사이클 동안 모든 캐쉬 라인들 중에서 드라우지 상태에 있는 캐쉬 라인의 비율을 의미한다. 즉, 드라우지 상태는 저전력 상태이므로 모든 캐쉬 라인들 중에서 드라우지 상태의 캐쉬 라인들의 비율이 높으면 높을수록 정적 캐쉬 에너지 소비는 줄어들게 됨을 의미한다. 이러한 척도는, 같은 정적 캐쉬 에너지 소비를 줄이기 위한 기법을 적용한 캐쉬라도 정적 에너지 소비는 gate length나 온도와 같은 요인에 의해서 상이한 결과가 나올 수 있기 때문에 사용하게 되었다. 이 척도는 gate length나 온도 등에 영향을 받지 않는다.

그림 8은 일정 프로세서 사이클(본 논문에서는 2000 사이클)동안 모든 캐쉬 라인들이 저전력 상태, 즉 드라우지 상태의 캐쉬 라인들일 때를 1로 봤을 때, 기본적인 드라우지 캐쉬와 제안되어진 캐쉬들이 상대적으로 얼마나 많은 비율의 드라우지 캐쉬 라인을 갖는 지를 정규화(normalized)한 결과를 보여준다. 64엔트리, 128엔트리, 256엔트리, 512엔트리 그리고 1024엔트리의 예측 테이블을 사용하였을 때, 평균적으로 각각 88.53%, 87.91%, 87.36%, 87.11%, 87.14%의 드라우지 상태의 캐쉬 라인들의 비율을 갖는다. 여기서 드라우지 상태의 캐쉬 라인 비율이 87% 이상이라는 의미는, 평균적으로 모든 캐쉬 라인들 중 87% 이상의 캐쉬 라인들이 매 일정 사이클 동안 접근이 한번도 이루어지지 않음을 의미한다. 그림 8에서 보면 각각의 벤치마크 프로그램에서 기본적인 드라우지 캐쉬보다는 정적 에너지 소비가 더 많음을 알 수 있다. 이것은 "드라우지 캐쉬 라인 미리 깨움"에서 예측 실패로 인한 잘못된 라인의 깨움으로 발생하는 정적에너지 증가를 의미한다.

4.3 동적 캐쉬 에너지 소비 감소

이번 절에서는 동적 캐쉬 에너지 소비의 감소에 대한 결과를 보인다. 그림 9는 제안되어진 통합 방식을 적용한 캐쉬의 동적 에너지 소비 감소에 대한 결과를 보이고 있다. 그래프에서 벤치 마크 프로그램 당 각 바(bar)는 일반적인 캐쉬(에너지 소비를 줄이기 위한 기법을 사용하지 않은 캐쉬)에서 소비되어지는 동적 에너지를 100으로 보았을 때, 1024, 512, 256, 128, 64엔트리의 예측 테이블 사용시 상대적 소비 에너지의 비율을 나타낸다. 또한 일반적인 드라우지 캐쉬는 어떠한 동적 캐쉬

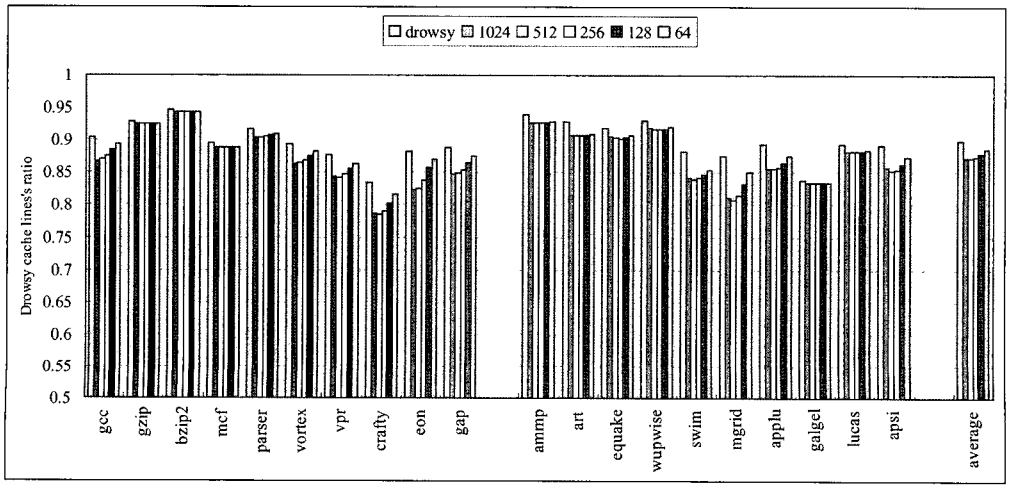

그림 8 정적 캐쉬 에너지 소비의 감소

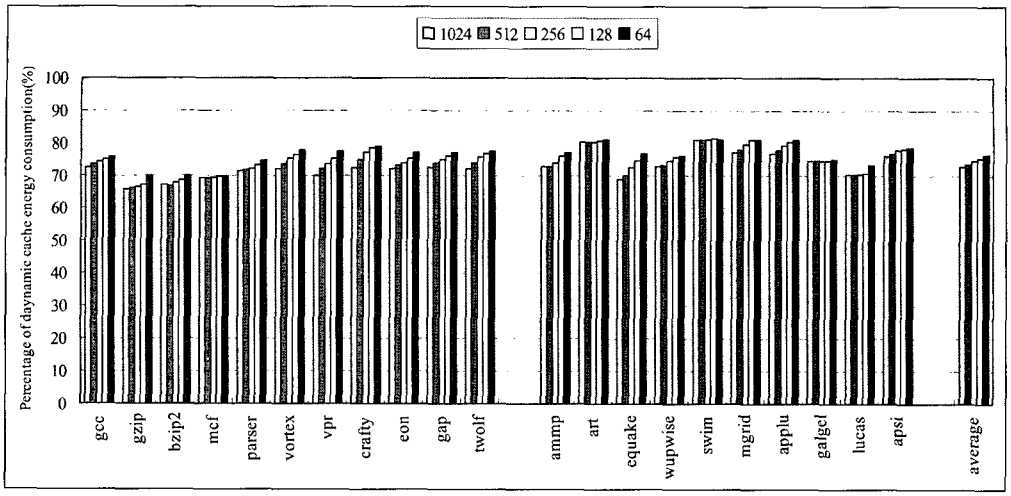

그림 9 동적 캐쉬 에너지 소비의 감소

에너지 소비를 감소시키기 위한 기법도 적용하고 있지 않기 때문에 위의 일반적인 캐쉬와 같다고 가정하였다.

1024, 512, 256, 128, 64엔트리의 예측 테이블을 이용할 때, 평균적으로 72.8%, 73.5%, 74.5%, 75.5%, 76.5%의 동적 캐쉬 에너지 소비를 보인다. 이는 달리말해서 기본적인 캐시에서 소비되어지는 동적 캐시 에너지 소비와 비교하여, 27.2%, 26.5%, 25.5%, 24.5%, 23.5%의 동적 캐쉬 에너지 소비의 감소를 의미한다.

동적 캐쉬 에너지 소비를 감소시키기 위해서 제안되어진 기법에서는 프로그램 카운터를 이용하는 웨이 예측 기법을 사용하고 있으며, 이미 이러한 프로그램 카운터를 이용하는 웨이 예측 기법에 대한 논문들에서 밝히고 있듯이, 예측 테이블의 크기가 웨이 예측의 정확성에 큰 영향을 끼치지 않으며, 웨이 예측 성공에 따른 동적 캐쉬 에너지 소비의 감소에도 큰 영향을 끼치지 못함을 나타낸다. 따라서 결과에서 보이는 바와 같이 예측 테이블의 엔트리 개수는 동적 캐쉬 에너지 소비 감소에 큰 영향을 끼치지 않고 있다.

4.4 성능 감소

캐쉬 에너지 소비 감소를 위해서 지금까지 제안되어진 대부분의 기법들이 에너지를 감소시키는 대신에 프로그램 실행 사이클의 증가를 가져오고 있다[1-5]. 본 논문에서 제안되어진 기법도 동적 캐쉬 에너지 소비를 줄이기 위해 프로그램 카운터를 이용하는 웨이 예측 기법의 사용과 정적 캐쉬 에너지 소비를 줄이기 위해 드라우지 캐쉬 기법을 사용으로 인해 성능 감소가 발생하게 된다. 이 성능 감소는 프로그램 카운터를 이용하는 웨이 예측의 실패로 인하여 발생하는 추가적인 프로그램 실행 사이클과, 거기에 더해서 드라우지 캐쉬 라인을 깨우기 위해 발생하는 추가적인 프로그램 실행 사이클에 기인한다. 즉, 제안 되어진 기법에서의 성능 감소는, ***추가 프로그램 실행 사이클 = 웨이 예측 실패에 따른 추가적인 프로그램 실행 사이클 + 드라우지 캐쉬 라인 깨움에 따른 추가적인 프로그램 실행 사이클***가 되며, 이러한 추가적인 프로그램 실행 사이클을 감소시키기 위하여 "드라우지 캐쉬 라인 미리 깨움"기법을 제안하고 적용하였으며, 4.1절에서 보이는 바와 같이, 드라우지 캐쉬 적용시 발생하는 추가적인 프로그램 실행 사이클 중, 드라우지 캐쉬 라인 깨움에 따른 추가적인 프로그램 실행 사이클을 감소시키고 있다.

그림 10과 그림 11은 기본적인 캐쉬를 적용한 시스템에 대한 드라우지 캐쉬만을 적용했을 때의 성능 감소와 "드라우지 캐쉬 라인 미리 깨움"을 적용하지 않은 웨이 예측 방식 + 드라우지 캐쉬 방식을 적용했을 때의 성능 감소, 그리고 "드라우지 캐쉬 라인 미리 깨움"을 적용한 웨이 예측 방식 + 드라우지 캐쉬 방식 적용에 따른 성능 감소를 보이고 있다.

그림 10에서 보이는 바와 같이 SPEC2000 INTEGER에서는 평균적으로 1.6%~2.2%의 성능 감소를 나타내고 있다. 그림 11의 SPEC2000 FP에서는 평균적으로 0.3%~0.45%의 성능 감소를 보이고 있다.

그림 10과 그림 11의 각 벤치마크 프로그램에서 가장 왼쪽의 막대는 드라우지 캐쉬만을 적용했을 때의 성능 감소를 나타낸다. 그림에서 x축의 1024, 512, 256, 128, 64 등은 웨이 예측 방식과 드라우지 캐쉬 방식만을 적용한 것을 의미하며, 1024+wakeup, 512+wakeup등은 웨이 예측과 드라우지 캐쉬 방식을 적용하고 거기에 더해서 "드라우지 캐쉬 라인 미리 깨움" 기법까지 같이 적용한 것을 의미한다. 그림의 각 바(bar)는 범례와 같이 동적 캐쉬 에너지 소비를 줄이기 위해 사용한 웨이

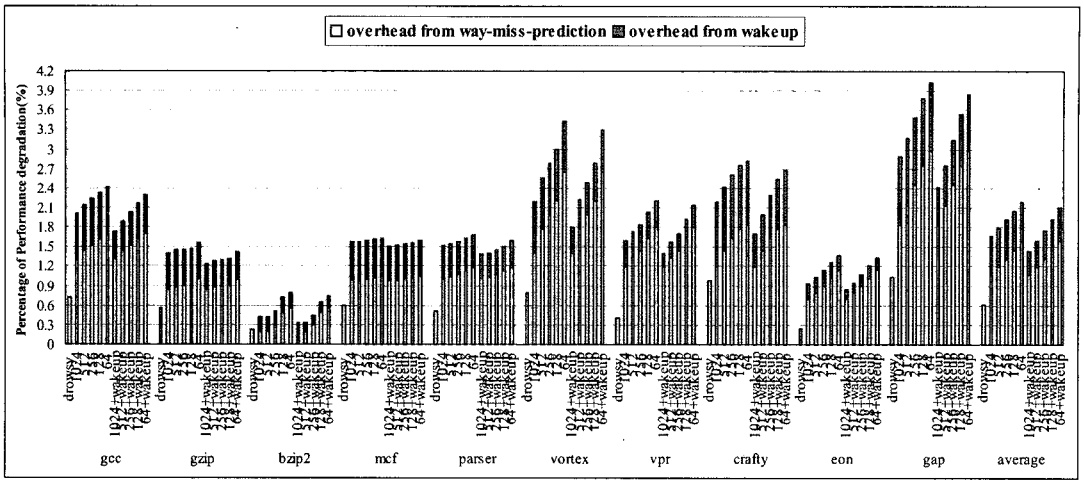

그림 10 성능 감소(SpecINT)

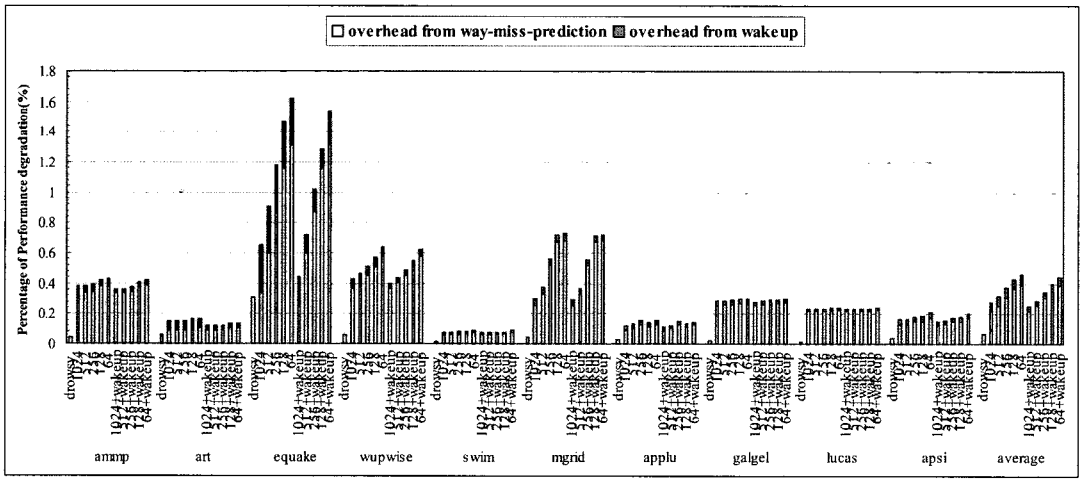

그림 11 성능 감소(SpecFP)

예측 기법에서 웨이 예측 실패로 인한 추가적인 프로그램 실행 사이클과 정적 캐쉬 에너지 소비를 줄이기 위해 사용한 드라우지 캐쉬 기법에서 깨움(wakeup)으로 발생하는 추가적인 프로그램 실행 사이클의 비율을 나타내고 있다. 그림 10에서 보이는 바와 같이 SPEC2000 INTEGER에 해당되는 벤치 마크 프로그램들에 대해서 전체의 추가적인 프로그램 실행 사이클 중, 웨이 예측 실패로 인해 발생하는 사이클의 비율이 평균적으로 80%에서 65%까지를 차지하고 있음을 알 수 있다. 즉 전체 추가적인 사이클에서 동적 캐쉬 에너지를 줄이기 위해 적용한 웨이 예측 기법의 예측 실패에 따른 추가 사이클의 비율이 큼을 보이며, 상대적으로 드라우지 캐쉬 적용에 따른 추가 사이클의 비율이 적음을 보이고 있다. 이는 그림 11의 SPEC2000 FP에 해당하는 벤치마크에서 그 비율이 더 커짐을 알 수 있는데, 이는 SPEC2000 FP에 해당하는 벤치마크 프로그램들의 특성에 기인한다.

위에서 설명된 것처럼 벤치마크 프로그램에서 드라우지 캐쉬 적용으로 발생하는 추가적인 실행 사이클 증가가 전체에서 차지하는 비율이 상대적으로 작기 때문에 "드라우지 캐쉬 라인 미리 깨움"을 적용했을 때 얻을 수 있는 사이클의 감소가 상대적으로 적다. 그림 10에서 "드라우지 캐쉬 라인 미리 깨움"기법을 적용하였을 때, 평균적으로 전체 성능 감소 대비 4.5%(64-entry)에서 14.1%(1024-entry)의 추가적인 사이클 감소 효과를 나타내고 있다. 또한 그림 11에서는 평균적으로 전체 성능 감소 대비 3.6%(64-entry)에서 11.7%(1024-entry)의 추가적인 사이클 감소 효과를 보이고 있다.

5. 결 론

에너지 소비의 감소는 고성능 임베디드 프로세서의 디자인에 있어서 중요한 이슈중 하나이며, 테크놀러지의 발전과 고성능 프로세서에 대한 요구로 인해 프로세서 내의 캐쉬 사이즈가 증가하고 있는 시점에 있어 더더욱 에너지 소비의 감소는 중요한 디자인 이슈가 되고 있다. 이러한 캐쉬 증가에 따른 에너지 소비의 감소를 위해 많은 기법들이 제안되어졌지만, 제안되어진 기법들은 정적 캐쉬 에너지와 동적 캐쉬 에너지의, 두 종류의 캐쉬 에너지 중 한 쪽의 캐쉬 에너지 소비를 줄이는 것에만 중점을 두어왔었다. 이에 본 논문에서는 정적 캐쉬 에너지와 동적 캐쉬 에너지 소비를 동시에 줄이는 통합 기법을 제안하였으며, 이 기법은 동적 에너지 소비를 줄이기 위해 기존에 제안되어진 프로그램 카운터를 이용하는 웨이 예측기법과 정적 에너지 소비를 줄이기 위한 드라우지 캐쉬 라인 기법을 통합한 기법이었다. 여기에 드라우지 캐쉬 기법을 적용함에 따라 생기는 추가적인 프로그램 실행 사이클을 감소시키기 위해, 예측 테이블에 기반한 "드라우지 상태의 캐쉬 라인 미리 깨움" 기법을 제안하여 추가적으로 적용시켰다. 본 논문에서 제안한 통합 기법과 미리 깨움 기법을 적용한 시스템에 대한 모의 실험 결과에서 보인바와 같이 "드라우지 캐쉬 라인 미리 깨움"에 의해서 일반적인 드라우지 캐쉬 라인에서 발생되는 성능 감소를 평균적으로 30% 감소시켰으며, 더불어 평균 25%의 동적 캐쉬 에너지 소비 감소와 86% 이상의 드라우지 상태 캐쉬 라인 비율(저전력 상태의 캐쉬 라인비율)을 갖게 됨으로써 그 만큼의 정적 캐쉬 에너지 소비 감소를 보이고 있다. 저전력을 구현하기 위한 기법 적용으로 생기는 성능 감소도 평균 1.5% 이하를 보이고 있다. 이는 지금까지의 저전력 캐쉬에 관계된 논문들과 달리 동적 캐쉬 에너지 소비와 정적 캐쉬 에너지 소비를 효과적으로 동시 감소시킬 수 있음을 보인 것이다.

참 고 문 헌

[1] K. Inoue, T. Ishihara, and K. Murakami., "Way-predicting set-associative cache for high performance and low energy consumption," In Proceedings of the International Symposium on Low Power Electronics and Design, pp. 273-275, August 1999.

[2] M. D. Powell, A. Agarwal, T. N. Vijaykumar, B. Falsafi and K. Roy. "Reducing set-associative cache energy via way-prediction and selective direct-mapping," In proceedings of international Symposium on Microarchitecture, December 2001.

[3] M. Powell, et. Al., "Gated-Vdd: A circuit technique to reduce leakage in deep-submission cache memories," In Proc. of Int. Symp. Low Power Electronics and Design, pp. 90-95, 2000.

[4] S. Kaxiras, Z. Hu and M. Martonosi., "Cache decay: Exploiting generational behavior to reduce leakage power," In Proc. International Symposium on Computer Architecture, July 2001.

[5] K. Flautner, N. S. Kim, S. Martin, D. Blaauw, and T. Mudge., "Drowsy caches: Simple techniques for reducing leakage power," In Proc. International Symposium on Computer Architecture, July 2002.

[6] J. L. Hennessy and D. A. Patterson, Computer Architecture: A Quantitative Approach, Second Edition, Morgan Kaufmann, 1996.

[7] Soontae Kim, N. Vijaykrishnan, M. J. Irwin and L. K. John., "On load latency in low-power caches," In Proc. of International Symposium Low Power Electronics and Design, 2003.

[8] D. Brooks, V. Tiwari, and M. Martonosi., "Wattch: A framework for architectural-level power analysis and optimizations," In Proceedings of the 27th Annual International Symposium on Computer Architecture, pages 8394, June 2000.

[9] D.C. Burger and T. M. Austin., The SimpleScalar Tool Set, Version 2.0, Computer Architecture News, 25 (3), pp. 13-25, June, 1997.

[10] SPEC CPU2000 Benchmarks, http://www.specbench.org.

심 성 훈

1997년 2월 동국대학교 전자계산학과 공학사. 1999년 2월 서울대학교 대학원 컴퓨터공학부 석사. 2000년 3월~현재 서울대학교 대학원 전기컴퓨터공학부 박사과정. 관심분야는 컴퓨터 구조, 저전력 시스템, 병렬처리 시스템

김 철 홍

1998년 2월 서울대학교 컴퓨터공학과 공학사. 2000년 2월 서울대학교 대학원 컴퓨터공학부 석사. 2000년 3월~현재 서울대학교 대학원 전기컴퓨터공학부 박사과정. 관심분야는 컴퓨터 구조, 저전력 시스템, 내장형 시스템, 병렬 처리

장 성 태

1986년 2월 서울대학교 전자계산기공학과 공학사. 1988년 2월 서울대학교 대학원 컴퓨터공학과 석사. 1994년 2월 서울대학교 대학원 컴퓨터공학과 박사. 1994년 3월~현재 수원대학교 IT대학 컴퓨터학과 부교수. 관심분야는 다중 프로세서 시스템, 병렬 처리, 캐쉬 구조, 저전력 임베디드 프로세서 설계, 모바일 시스템

전 주 식

1975년 2월 서울대학교 응용수학과 학사 1977년 2월 한국과학기술원 전산학과 석사. 1983년 2월 미국 Univ. of Utah 박사. 1983년~1985년 Univ. of Iowa 조교수. 1985년~현재 서울대학교 전기컴퓨터공학부 교수. 관심분야는 컴퓨터 구조, 병렬 처리, VLSI/CAD