韓國 컴퓨터情報學會 論文誌
第13卷 第6號, 2008. 11.

2008-13-6-1-4

임베디드 시스템에서 후방 분기 명령어 정보를 이용한 저전력 명령어 캐쉬 설계 기법

양 나 라*, 김 종 면**, 김 철 홍***

Energy-aware Instruction Cache Design using Backward Branch Information for Embedded Processors

Na Ra Yang*, Jong Myon Kim**, Cheol Hong Kim***

요 약

반도체 기술의 급속한 발달과 함께 임베디드 프로세서의 성능이 점차 강력해지면서 몇 가지 문제점이 발생하게 되었다. 그 중에서도 프로세서 내에서 소비되는 에너지의 급격한 증가는 심각한 문제이다. 이러한 이유로 인해 최신의 임베디드 프로세서를 설계할 때에는 성능과 함께 에너지 효율성이 반드시 고려되어야 한다. 본 논문에서는 프로세서에서 소비되는 에너지의 상당 부분을 차지하고 있는 명령어 캐쉬의 에너지 효율성을 향상시키기 위해 후방 분기 명령어 정보를 이용하는 기법을 제안하고자 한다. 큰 크기의 주 명령어 캐쉬와 작은 크기의 순환문 캐쉬로 구성되는 제안된 기법을 통해 프로세서의 요청이 올 때 주 명령어 캐쉬와 순환문 캐쉬 중에서 하나의 캐쉬만이 선택적으로 접근되도록 하여 주 명령어 캐쉬의 접근 횟수를 크게 감소시킴으로써 우수한 에너지 효율성을 얻을 수 있다. 실험 결과, 제안하는 저전력 명령어 캐쉬는 기존의 명령어 캐쉬와 비교하여 평균 20%의 에너지 소비를 감소시킨다는 사실을 확인하였다.

Abstract

Energy efficiency should be considered together with performance when designing embedded processors. This paper proposes a new energy-aware instruction cache design using backward branch information to reduce the energy consumption in an embedded processor, since instruction caches consume a significant fraction of the on-chip energy. Proposed instruction cache is composed of two caches: a large main instruction cache and a small loop instruction cache. Proposed technique enables the selective access between the main instruction cache and the loop instruction cache to reduce the number of accesses to the main instruction cache, leading to good energy efficiency. Analysis results show that the proposed instruction cache reduces the energy consumption by 20% on the average, compared to the traditional instruction cache.

▸ Keyword : Embedded System(임베디드 시스템), Processor Architecture(프로세서 구조), Low-power System(저전력 시스템), Instruction Cache(명령어 캐쉬)

• 제1저자 : 양나라 교신저자 : 김철홍
• 접수일 : 2008. 8. 20, 심사일 : 2008. 11. 8, 심사완료일 : 2008. 11. 26.
* 전남대학교 전자컴퓨터공학과 석사과정 **울산대학교 컴퓨터정보통신공학부 교수
*** 전남대학교 전자컴퓨터공학부 교수
※ 이 논문은 2007년도 전남대학교 학술연구비 지원에 의하여 연구되었음.

Ⅰ. 서 론

기존의 임베디드 프로세서 (Embedded Processor)를 설계하는데 있어서는 칩 공간 (Chip Area)을 줄이는 기법과 성능을 향상시키는 기법이 설계의 주된 초점이 되어 왔다. 하지만, 임베디드 프로세서의 성능이 점차 강력해짐에 따라 프로세서 내에서 소모되는 에너지가 크게 증가하는 문제가 발생하게 되었다. 소모되는 에너지가 증가하면 배터리 사용 시간이 감소하고 칩의 온도가 높아지면서 쿨링 비용 (Cooling Costs)이 커지는 심각한 문제가 발생한다. 그러므로 최신의 임베디드 프로세서를 설계하는데 있어서는 성능 못지 않게 에너지 효율성이 중요하게 고려되고 있다.

프로세서에서 소비되는 에너지를 감소시키기 위한 기법은 크게 회로 수준 (Circuit-level)에서의 에너지 감소 기법과 구조적 수준 (Architecture-level)에서의 에너지 감소 기법으로 분류할 수 있다. 예전에는 주로 회로 수준에서의 기법을 통해 프로세서 내에서 소비되는 에너지를 감소시키기 위한 연구가 주로 이루어졌다 [1][2]. 하지만, 회로 수준의 에너지 감소 기법만으로는 한계에 다다르게 되었고, 이를 해결하기 위해 최근에는 구조적 수준에서의 접근 방식을 통해 더욱 우수한 에너지 최적화 결과를 얻기 위한 연구가 활발하게 이루어지고 있다 [3][4].

프로세서 설계 시 일반적으로 사용되고 있는 계층적 캐쉬 메모리 구조에서 1차 명령어 캐쉬는 거의 모든 사이클마다 접근이 이루어지므로 캐쉬 메모리들 중에서도 상당히 많은 에너지를 소모하는 모듈이다 [5]. 그러므로, 프로세서의 에너지 효율성은 1차 명령어 캐쉬의 에너지 효율성과 큰 상관 관계를 가지게 된다. 본 논문에서는 후방 분기 명령어 정보를 활용하여 성능 저하를 최소화하면서 명령어 캐쉬에서 소비되는 에너지를 크게 감소시키는 기법을 제안하고자 한다. 제안하는 기법에서는 후방 분기 명령어 정보를 이용하여 항상 주 명령어 캐쉬에 접근하는 대신 순환문 (Loop)이 수행되는 경우에는 주 명령어 캐쉬에 비해 훨씬 크기가 작은 순환문 캐쉬에 접근하게 함으로써 소모되는 에너지를 감소시키고자 한다.

이하 본 논문의 구성은 다음과 같다. II장에서는 관련 연구들을 통해 기존의 연구내용을 분석한다. III장에서는 순환문이 수행되는 경우 후방 분기 명령어 정보를 활용하여 에너지 소비를 줄일 수 있는 방식을 제시하고, 이를 위하여 설계된 캐쉬 구조를 설명한다. IV장에서는 본 논문에서 제안한 캐쉬 구조의 에너지 효율성을 측정하기 위한 모의 실험 방법과 실험 결과를 기술한다. 끝으로 V장에서 결론을 맺는다.

II. 관련 연구

프로세스에서의 에너지 소비를 감소시키기 위한 연구 중 상당 수는 캐쉬 (Cache) 메모리를 목표로 이루어지고 있다. 이는 캐쉬 메모리가 프로세서 내에서 소모되는 에너지의 상당 부분을 차지하고 있기 때문이다 [6]. 이러한 이유로, 캐쉬의 에너지 효율성을 높이는 기법을 통해 임베디드 프로세서 내의 에너지 소모를 줄이기 위한 연구들도 지속적으로 이루어지고 있다.

캐쉬에서 소모되는 에너지를 감소시키기 위한 대표적인 기법 중 하나인 필터 캐쉬 (Filter Cache) 기법은 작은 크기의 캐쉬 모듈을 프로세서 코어와 주 캐쉬 (Main Cache) 사이에 삽입함으로써 주 캐쉬로의 접근 횟수를 감소시킴으로써 에너지 소모를 줄이는 방법이다 [7]. 필터 캐쉬 기법은 주 캐쉬의 활성화 횟수를 줄임으로써 동적 에너지를 줄이기 위한 대표적인 기법으로, 필터 캐쉬의 에너지 효율성을 보다 높이기 위한 방안과 동적 에너지 뿐 아니라 정적 에너지를 줄이기 위한 기법도 제안되었다 [8][9]. 큰 크기의 주 명령어 캐쉬와 프로세서 코어 사이에 작은 크기의 캐쉬를 삽입한 후, 컴파일러와의 연동을 통해 캐쉬 내의 에너지 소비를 줄이기 위한 연구 또한 제안되었다 [10].

캐쉬에서 소모되는 에너지는 주로 캐쉬의 크기나 연관성 (Associativity)과 같은 캐쉬의 구성에 크게 좌우된다. 일반적으로, 한 번 접근 시 소모되는 동적 에너지는 캐쉬의 크기가 커질수록 증가하게 된다. 하지만, 캐쉬의 크기가 작게 되면 미스율 (Miss Rates)의 증가로 인해 프로세서의 성능이 저하되므로 많은 에너지 소모에도 불구하고 크기가 큰 캐쉬를 일반적으로 사용하게 된다. 이와 같은 사실을 활용하여, 캐쉬에 대한 접근이 많지 않은 시간에는 연관캐쉬 (Set-associative Cache) 내의 일부 웨이들 (Way)을 동적으로 비활성화시키고, 캐쉬 접근이 많은 시간에만 모든 웨이들을 활성화시키는 기법을 통해서 캐쉬의 에너지 소모를 줄이는 기법도 제안되었다 [11]. 캐쉬로의 접근 요청이 있을 때 요청되는 데이터가 있을 것으로 예측되는 특정 웨이만을 먼저 접근하고, 이 예측이 틀린 경우에만 다른 웨이들을 접근함으로써 약간의 성능 저하를 통해 큰 에너지 감소 효과를 얻는 기법 또한 제안되었다 [12]. 활성화되는 캐쉬의 크기를 줄이기 위해 구조적으로 캐쉬를 여러 개의 부캐쉬들로 분할함으로써 캐쉬에서 소비되는 에너지를 감소시키는 기법도 제안된 바 있다 [13].

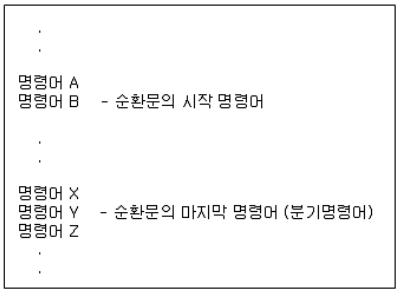

그림 1. 명령어 흐름 내에서 순환문의 예
Fig. 1. Loop Example in the Instruction Stream

III. 저전력 명령어 캐쉬 구조

1. 후방 분기 명령어

상당 수의 임베디드 응용 프로그램 내에서 순환문 내의 명령어들은 수행되는 총 명령어의 많은 부분을 차지하고 있다. 일반적으로 하나의 순환문은 여러 개의 명령어들로 구성된다. 그림 1은 응용 프로그램 내에서 수행되는 순환문의 예를 보여준다. 그림에서 보이는 순환문은 명령어 B에서부터 시작한다. 명령어 Y는 순환문의 마지막 명령어로서 이후의 명령어 흐름을 결정하는 분기 명령어이다. 이와 같은 경우에, 만약 분기 명령어 Y가 언테이큰 (Untaken)된다면, 명령어 Y 이후에 수행되는 명령어는 명령어 Z가 된다. 이와 달리, 분기 명령어 Y가 테이큰 (Taken)된다면, 명령어 Y 이후에는 다시 순환문의 시작 명령어인 명령어 B가 수행되고, 이 경우 순환문 내의 명령어들은 (명령어 B부터 명령어 X까지) 재차 수행되게 된다. 이와 같이, 순환문의 마지막에 위치하여 후방 분기를 통해 명령어의 흐름을 다시 순환문의 시작 부분으로 이동시키는 명령어를 후방 분기 명령어라고 한다. 즉, 순환문의 마지막 명령어인 분기 명령어가 후방 분기 명령어인 경우에는 이미 수행된 순환문 내의 명령어들이 재차 수행되게 되므로, 동일한 명령어들로 인하여 명령어 캐쉬가 반복적으로 접근되는 상황이 발생하게 된다.

본 논문에서는 이와 같이 명령어 흐름 내에서 후방 분기 명령어가 수행되는 경우 동일한 명령어들에 대한 반복적인 명령어 캐쉬 접근으로 인해 소비되는 에너지를 감소시키기 위하여 새로운 캐쉬 구조를 제안한다.

2. 제안하는 저전력 명령어 캐쉬 구조

후방 분기 명령어가 실행되는 경우에 순환문 내의 동일한 명령어들이 반복적으로 명령어 캐쉬에서 인출되면서 소비되는 에너지를 줄이기 위해 순환문 캐쉬 (SLC: Small Loop Cache)를 본 논문에서는 제안한다. 순환문 캐쉬는 순환문 내의 비분기 명령어들을 저장하기 위한 작은 크기의 캐쉬이다.

순환문 캐쉬를 이용한 저전력 명령어 캐쉬 구조는 그림 2에서 보이는 바와 같다. 제안하는 저전력 명령어 캐쉬는 큰 크기의 주 명령어 캐쉬와 작은 크기의 순환문 캐쉬로 구성된다. 한 번 접근 시 소비되는 에너지는 주 명령어 캐쉬에 비해 순환문 캐쉬가 훨씬 적을 것으로 예상된다. 순환문 캐쉬의 크기가 주 명령어 캐쉬와 비교하여 훨씬 더 작기 때문이다. 그러므로 제안된 방식을 통해 주 명령어 캐쉬에 대한 접근을 순환문 캐쉬로의 접근으로 대체한다면 에너지 소비를 감소시키게 된다.

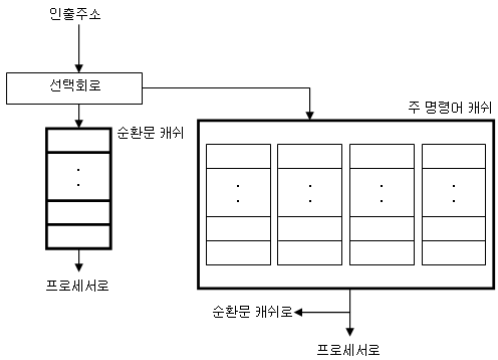

그림 2. 제안하는 저전력 명령어 캐쉬 구조
Fig. 2. Proposed Energy-aware Instruction Cache

그림 2에서의 선택회로는 직전에 수행된 후방 분기 명령어의 주소값을 저장하고 있는 레지스터 (Register), 순환문 내의 명령어 개수를 저장하기 위한 레지스터, 비교기 (Comparator), 그리고 카운터 (Counter)로 구성된다. 후방 분기 명령어가 수행되는 경우 수행된 후방 분기 명령어의 주소는 선택회로에 저장된다. 이후에 명령어가 수행될 때에는 저장되어 있는 후방 분기 명령어의 주소값과 수행되는 명령어의 주소가 일치하는지를 살펴봄으로써 동일한 후방 분기 명령어가 재수행되는 경우를 찾아낼 수 있다. 후방 분기 명령어가 재수행되는 경우에는 카운터를 이용하여 재차 수행되는 순환문 내의 명령어 개수를 세어 그 값을 선택회로에 저장한다. 이와 동시에, 주 명령어 캐쉬에서 인출되는 순환문 내의 명령어들은 프로세서 코어로 전송될 때 순환문 캐쉬로도 전송이 이루어진다. 이후에, 동일한 후방 분기 명령어가 다시 수행된

다면 저장된 순환문 내의 명령어 개수를 활용하여 선택회로가 순환문 내의 명령어 인출을 위한 주 명령어 캐쉬로의 접근을 순환문 캐쉬로의 접근으로 대체시킨다.

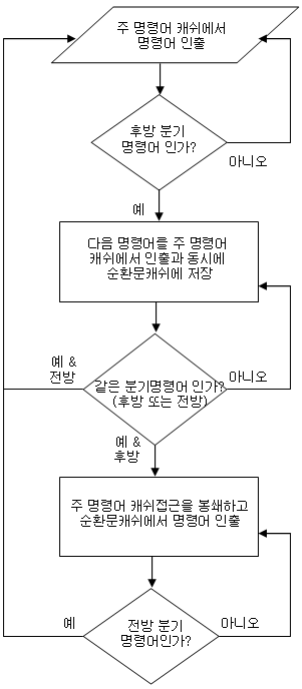

그림 3. 제안하는 저전력 명령어 캐쉬의 제어 흐름도
Fig. 3. Control Flow for the Proposed Instruction Cache

제안하는 저전력 명령어 캐쉬 구조의 동작을 정리하면 그림 3에서 보이는 바와 같다. 일반적인 상태에서는 제안한 저전력 명령어 캐쉬로 프로세서가 명령어를 요청하면 주 명령어 캐쉬로의 접근이 이루어진다. 명령어 흐름에서 후방 분기 명령어를 만나게 되면 선택회로는 후방 분기 명령어의 주소값을 저장한다. 동일한 후방 분기 명령어가 재차 수행되면 순환문 내의 명령어들을 프로세서 코어로 전송할 때 순환문 캐쉬로도 명령어를 전송하여 저장하고, 순환문 내의 명령어의 개수 또한 계산하여 저장한다. 이후, 동일한 후방 분기 명령어가 다시 수행되는 경우에는, 순환문 내의 명령어 개수만큼 주 명령어 캐쉬로의 접근은 봉쇄되고, 대신 순환문 내의 명령어들은 순환문 캐쉬에서 인출된다. 주 명령어 캐쉬로의 접근은 순환문의 마지막 명령어가 테이큰 되지 않을 때까지, 즉 전방 분기 명령어로 판별될 때까지 봉쇄된다.

기존의 명령어 캐쉬에서는 프로세서로부터 명령어 요청이 있을 때마다 주 명령어 캐쉬가 활성화되어 명령어를 인출한다. 그 결과, 많은 에너지가 소모되었다. 반면 제안한 명령어 캐쉬에서는 순환문 내의 동일 명령어들이 반복적으로 수행될 때에는 주 명령어 캐쉬로의 접근이 봉쇄되고 대신 순환문 캐쉬가 활성화되어 명령어들이 인출된다. 이를 통해, 큰 크기의 주 명령어 캐쉬로의 접근을 작은 크기의 순환문 캐쉬로의 접근으로 대체함으로써 상당 부분의 에너지 소비를 감소시킬 수 있을 것으로 예상된다.

IV. 모의 실험

이 장에서는 제안한 저전력 명령어 캐쉬의 에너지 효율성을 측정하기 위한 실험 방법을 설명하고 그 결과를 제시하고자 한다. 본 연구에서는 에너지 효율성 측정을 위하여 SimpleScalar [14] 시뮬레이터를 수정하여 모의 실험을 하였고, 모의 실험 시 전력 변수들은 CACTI [15]로부터 얻었다. 모의 실험 대상으로는 ARM 임베디드 프로세서와 유사한 구조의 2-way superscalar 프로세서를 사용하였다 [16]. 모의 실험 시 사용한 시스템 구성 변수들은 표 1에서 보이는 바와 같다. 표에 나타낸 바와 같이, 순환문 캐쉬의 크기는 64Bytes, 128Bytes, 256 Bytes의 세 종류를 사용하였다. 모의 실험을 위한 벤치마크 프로그램으로는 SPEC CPU 2000 [17] 프로그램들을 사용하였다.

표 1. 모의 실험 시 시스템 구성 변수
Table 1. System Configuration Parameters

실험인자	값
Fetch, Decode, Issue, Commit	2 instructions/cycle
Functional Units	2 integer ALUs, 2 FP ALUs, 1 integer multiplier/divider, 1 FP multiplier/divider
L1 I-Cache	32KB, 32-way, 32bytes lines, 1 Cycle latency
L1 D-Cache	32KB, 32-way, 32bytes lines, 1 Cycle latency
SLC	64B ~ 256B, fully-associative, 32 byte lines, 1 cycle latency
RUU size	16
LSQ size	8
Branch Predictor	Bimodal

1. 주 명령어 캐쉬에 접근하는 횟수

모의 실험을 통해 얻은 주 명령어 캐쉬의 접근 횟수는 그림 4에 보이는 바와 같다. 그래프에서 Trad는 순환문 캐쉬가 없는 기존의 명령어 캐쉬를 나타내며, SLC64, SLC128, SLC256은 각각 64Bytes, 128Bytes, 256Bytes 크기의 순환문 캐쉬를 포함하는 제안한 캐쉬 구조를 나타낸다.

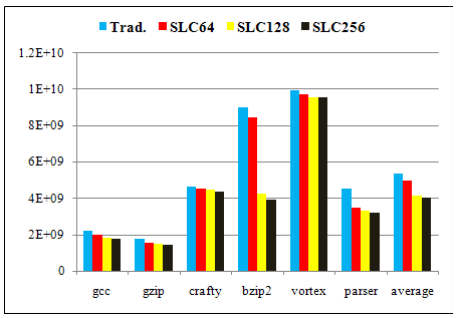

그림 4. 주 명령어 캐쉬 접근 횟수
Fig. 4. Number of Access to the Main Instruction Cache

그래프에 보이는 바와 같이 제안된 기법은 주 명령어 캐쉬로의 접근 횟수를 크게 감소시킨다. 주 명령어 캐쉬의 접근 횟수는 순환문 캐쉬의 크기가 커질수록 더욱 감소된다는 것을 알 수 있다. 순환문 캐쉬를 활용한 제안하는 캐쉬 구조는 실행되는 총 명령어 중 순환문이 많은 부분을 차지하는 프로그램일 수록 더욱 많은 주 명령어 캐쉬로의 접근 횟수를 감소시킨다. 예를 들어, 순환문에 포함된 명령어가 많은 bzip2에서, SLC256은 기존 캐쉬에 비하여 주 명령어 캐쉬로의 접근 횟수를 56% 감소시킨다. 여기서 주목할 점은 주 명령어 캐쉬에 대한 접근 횟수를 줄이는 것은 에너지 효율성을 향상시킨다는 사실이다.

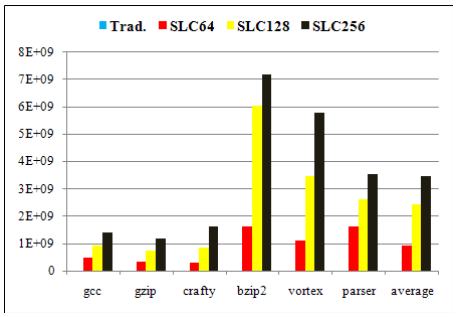

그림 5. 순환문 캐쉬 접근 횟수
Fig. 5. Number of Access to the SLC

그림 5는 모의실험을 통한 각 캐쉬 구조에 따른 순환문 캐쉬로의 접근 횟수를 나타낸다. 그림 4와 그림 5를 통해 감소된 주 명령어 캐쉬로의 접근은 순환문 캐쉬로의 접근으로 대체되었음을 알 수 있다. 보다 많은 수의 주 명령어 캐쉬 접근이 대체될수록 높은 에너지 효율성을 나타낼 것으로 기대된다.

2. 에너지 효율성

표 2는 CACTI로부터 얻어진 각 캐쉬 모델들의 한 번 접근에 따른 에너지 소비를 보여준다. 예상한 바와 같이, 순환문 캐쉬의 한 번 접근에 따른 에너지 소비가 주 명령어 캐쉬의 한 번 접근 시 소비되는 에너지에 비해 훨씬 적다는 사실을 확인할 수 있다.

표 2. 한번 접근 시 소모되는 에너지
Table 2. Per-access Energy Consumption

캐쉬모델	한번 접근할 때 소모되는 에너지(nJ)
주 명령어 캐쉬	3.70756
64Bytes 순환문캐쉬	0.13248
128Bytes 순환문 캐쉬	0.13436
256Bytes 순환문 캐쉬	0.13813

각 응용 프로그램 수행 시 소비된 에너지에 대한 비교 결과는 그림 6에서 보여준다. 그래프에 표시한 에너지 소비는 명령어가 인출 되는 동안 명령어 캐쉬에서 소비된 동적 에너지의 총 합을 나타내고 결과는 기존 명령어의 에너지 소비로 정규화하여 표시하였다. 제안하는 기법의 효율성을 보다 명확하게 알아볼 수 있도록 필터 캐쉬를 응용한 기법인 디코드 필터 캐쉬 [8]와도 에너지 효율성을 비교해 보았다. 그래프에서 DFC는 디코드 필터 캐쉬를 나타낸다.

실험결과를 통해 SLC256은 기존의 캐쉬와 비교하여 명령어를 인출할 때 소비되는 에너지를 평균 20% 감소시킨다는 사실을 확인하였다. 이러한 에너지 소비의 감소는 주 명령어 캐쉬에 대한 접근을 순환문 캐쉬에 대한 접근으로 대체시킴으로써 얻어졌다. 그래프에서 보이는 바와 같이, DFC와 비교해서도 SLC128, SLC256은 평균적으로 보다 우수한 전력 효율성을 보이고 있다.

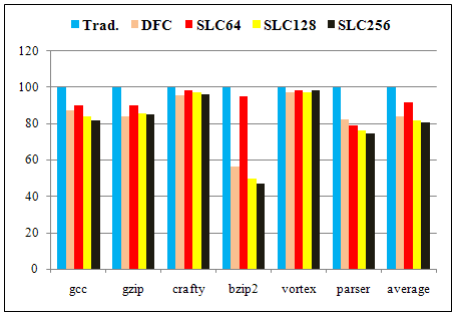

그림 6. 캐쉬 구성에 따른 에너지 소비량 비교
Fig. 6. Normalized Energy Consumption

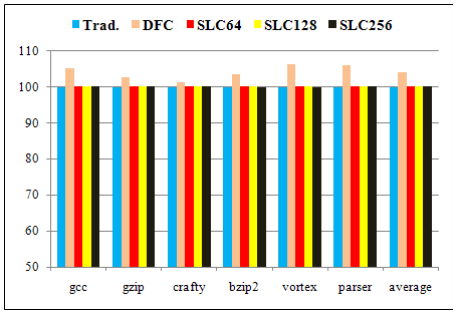

그림 7. 캐쉬 구성에 따른 실행 시간 비교
Fig. 7. Normalized Execution Time

그림 7은 각 캐쉬 구조에 따른 성능을 나타낸다. 그래프에서 보이는 바와 같이 제안하는 구조는 프로세서의 성능에는 거의 영향을 미치지 않는다는 것을 알 수 있다. 이에 반해 디코드 필터 캐쉬 기법은 성능 저하가 발생하는 것을 알 수 있다. 제안하는 구조는 주 명령어 캐쉬에 대한 접근을 순환문 캐쉬에 대한 접근으로 대체시키는 것이기 때문에 디코드 필터 캐쉬와는 달리 캐쉬의 미스율에는 아무런 영향을 주지 않는다는 장점을 가진다는 것을 확인할 수 있다.

V. 결 론

본 논문에서는 순환문 캐쉬를 사용하여 주 명령어 캐쉬에 대한 접근 횟수를 줄이는 새로운 명령어 캐쉬 구조를 제안하였다. 제안한 기술은 주 명령어 캐쉬에 매번 접근하던 기존의 구조와 달리 주 명령어 캐쉬와 순환문 캐쉬 사이에서 후방 분기 명령어 정보를 이용하여 선택적으로 접근이 가능하도록 하였다. 이를 통해, 순환문 내의 명령어들이 반복적으로 인출되는 경우 큰 크기의 주 명령어 캐쉬에 대한 접근을 작은 크기의 순환문 캐쉬에 대한 접근으로 대체 하였다. 이 기법을 통해, 큰 크기의 주 명령어 캐쉬에만 계속적으로 접근하던 기존의 캐쉬와 비교하였을 때 상당한 에너지 소비를 감소시킬 수 있었다. 그에 반해 성능 감소는 거의 발생하지 않았다. 그러므로 에너지 효율성을 크게 향상시킨 제안된 명령어 캐쉬 구조는 향후 저전력 임베디드 프로세서를 위한 좋은 해결책 중 하나로 기대된다.

참고문헌

[1] K. Ghose and M. B. Kamble, "Reducing Power in Superscalar Processor Caches using Subbanking, Multiple Line Buffers and Bit-line Segmentation", In Proceedings of International Symposium on Low Power Electronics and Design, pages 70 - 75, 1999.

[2] M. B. Kamble and K. Ghose, "Analytical Energy Dissipation Models for Low-Power Caches", In Proceedings of International Symposium on Low Power Electronics and Design, pages 143 - 148, 1997.

[3] A. Ma, M. Zhang, and K. Asanovic. "Way Memorization to Reduce Fetch Energy in Instruction Caches", In Workshop on Complexity-Effective Design, 28th ISCA, 2001.

[4] E. Witchel and K. Asanovic, "The Span Cache: Software Controlled Tag Checks and Cache Line Size", In Workshop on Complexity-Effective Design, 28th ISCA, 2001.

[5] J. Montanaro et al, "A 160 Mhz, 32b, 0.5W CMOS RISC Microprocessor", In Proceedings of International Solid-State Circuits Conference, pp. 214-229, 1996.

[6] S. Segars, "Low Power Design Techniques for Microprocessors", In Proceedings of International Solid-State Circuits Conference, 2001.

[7] J. Kin, M. Gupta, and W. Mangione-Smith, "The Filter Cache: An Energy Efficient Memory Structure", In Proceedings of International Symposium on Microarchitecture, pp. 184-193, 1997.

〔8〕 K. Vivekanandarajah, T. Srikanthan, and S. Bhattacharyya, "Decode Filter Cache for Energy Efficient Instruction Cache Hierarchy in Superscalar architectures", In Proceedings of 2004 Conference on Asia South Pacific Design Automation, pp. 373-379, 2004.

〔9〕 R. Giorgi and P. Bennati, "Reducing Leakage in Power-saving Capable Caches for Embedded Systems by Using a Filter Cache", In Proceedings of the 2007 Workshop on Memory Performance: Dealing with Applications, Systems, and Architecture, pp. 97-104, 2007.

〔10〕 N. Bellas, I. Hajj, and C. Polychronopoulos, "Using Dynamic Cache Management Techniques to Reduce Energy in a High-performance Processor", In Proceedings of International Symposium on Low Power Electronics and Design, pp. 64-69, 1999.

〔11〕 David H. Albonesi, "Selective Cache Ways: On Demand Cache Resource Allocation", In Proceedings of International Symposium on Microarchitecture, pp. 70-75, 1999.

〔12〕 M. 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, pp. 54-65, 2001.

〔13〕 C. H. Kim, S. W. Chung, and C. S. Jhon, "PP-cache: A Partitioned Power-aware Instruction Cache Architecture", Microprocessors and Microsystems, Vol. 30, Issue 5, pp. 268-279, 2006.

〔14〕 D. Burger, T. M. Austin, and S. Bennett, "Evaluating Future Microprocessors: the Simplescalar tool set", Technical Report TR-1308, Univ. of Wisconsin-Madison Computer Sciences Dept, 1997.

〔15〕 P. Shivakumar and N. P. Jouppi, "Cacti 3.0: An Integrated Cache Timing, Power, and Area Model", Technical Report TRWRL-2001-2, Univ. of Wisconsin-Madison Computer Sciences Dept, 2001.

〔16〕 ARM. ARM1136J(F)-S, http://www.arm.com/products/CPUs/ARM1136JF-S.html.

〔17〕 SPEC. SPEC CPU2000 Benchmarks http://www.specbench.org.

저 자 소 개

양 나 라

- 2008년 호남대학교 컴퓨터공학과 학사
- 2008년 전남대학교 전자컴퓨터공학부 석사과정 입학
- 관심분야: 컴퓨터구조, 임베디드시스템

김 종 면

- 1995년 명지대학교 전기공학사
- 2000년 University of Florida 석사
- 2005년 Georgia Institute of Technology 박사
- 2005년~2007년 삼성종합기술원 전임연구원
- 2007년~현재 울산대학교 컴퓨터정보 통신공학부 교수
- 관심분야: 임베디드 SoC, 컴퓨터구조, 프로세서 설계, 병렬처리

김 철 홍

- 1998년 서울대학교 컴퓨터공학사
- 2000년 서울대학교 컴퓨터공학부 석사
- 2006년 서울대학교 전기컴퓨터공학부 박사
- 2005년~2007년 삼성전자 반도체총괄 책임연구원
- 2007년~현재 전남대학교 전자컴퓨터공학부 교수
- 관심분야: 임베디드시스템, 컴퓨터구조, SoC 설계, 저전력 설계