Journal of The Korea Society of Computer and Information
Vol. 18, No. 9, September 2013
www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2013.18.9.011

지역 버퍼와 주소 압축을 통한 저전력 캐시 설계

곽 종 욱*

Low-Power Cache Design by using Locality Buffer and Address Compression

Jong Wook Kwak*

요 약

프로세서와 메모리 시스템 사이의 속도 차이를 완화하기 위하여 오늘날의 컴퓨터 시스템은 대부분 캐시 시스템을 사용하고 있다. 하지만 소비 전력 측면에서 캐시 메모리는 전체 시스템 측면에서 큰 비중을 차지한다. 본 논문에서는 캐시 시스템의 전력을 줄이는 방안 가운데 하나로 지역 버퍼와 주소 압축을 통한 저전력 캐시 설계 기법을 제안한다. 주소 압축을 위해 사용되는 부분태그 캐시는 전력 소모량을 최소화하기 위해서 전체 태그를 쓰기보다는 태그의 작은 부분을 사용함으로써 소비 전력을 줄이도록 하는 기법이다. 본 논문에서는 기존의 여러 주소 압축 캐시 연구에서의 문제점들을 분석하여 그것을 보완할 수 있는 새로운 기법을 제안한다. 제안된 기법은 지역성이 높은 내장형 응용프로그램의 특징을 활용한 것으로, 지역 버퍼와 지역 실패 버퍼를 활용한 새로운 형태의 캐시 주소 압축 기법이다. 모의실험 결과, 제안된 기법은 전체적인 성능의 감소 없이 평균 18%의 에너지 감소를 보였다.

▸ Keywords : 저전력 캐시, 부분태그 캐시, 지역성, 캐시 태그, 주소 압축

Abstract

Most modern computer systems employ cache systems in order to alleviate the access time gap between processor and memory system. The power dissipated by the cache systems becomes a significant part of the total power dissipated by whole microprocessor chip. Therefore, power reduction in the cache system becomes one of the important issues. Partial tag cache is the system for the least power consumption. The main power reduction for this method is due to the use of small partial tag matching, not full tag matching. In this paper, we first analyze the previous regular partial tag cache systems and propose a new address matching mechanism by using locality buffer and address compression. In simulation results, the proposed model shows 18% power

•제1저자 : 곽종욱 •교신저자 : 곽종욱
•투고일 : 2013. 5. 5, 심사일 : 2013. 6. 27, 게재확정일 : 2013. 7. 14.
* 영남대학교 컴퓨터공학과(Dept. of Computer Engineering, Yeungnam University)

reduction in average, still providing same performance level, compared to regular cache.

▸ Keywords : Low power cache, Partial tag cache, Locality, Cache tag, Address compression

I. 서 론

빠르게 발전하고 있는 초고밀도집적회로 기술은 프로세서의 처리 속도를 급격하게 높여왔다. 이와 함께, 명령어 수준의 병렬처리(ILP) 기술의 발전과 클락 속도의 증가는 처리 속도의 증가를 가속화 시켜 프로세서의 성능을 매년 60% 이상 향상시켜 왔다. 그러나 컴퓨터 시스템 상에서 또 하나의 중요한 한 축을 이루는 메모리의 동작 속도는 크게 증가하지 않았다. 메모리 가운데 가장 많이 쓰이는 동적 램(DRAM)은 매년 10% 미만의 속도 증가만을 보여주고 있다. 이러한 프로세서와 메모리 시스템간의 큰 속도 차이는 곧 시스템 성능 향상의 주된 저해 요인으로 작용한다[1].

캐시 메모리 시스템은 이런 단점을 보완하기 위하여, 동작 속도 면에서 프로세서보다는 느리지만, 동적 램보다는 빠른 정적 램(SRAM)을 프로세서와 메모리 시스템 사이에 추가하여 이들 상호간의 속도 차이를 절충해 주었다. 최신 마이크로프로세서들은 대부분 레벨 1 혹은 레벨 2 캐시 시스템, 더 나아가 레벨 3 캐시 시스템까지 채택하고 있다. 이것은 전체적인 시스템 성능의 큰 발전을 가져왔지만, 칩에서 적지 않은 영역을 차지하게 되어 결과적으로 캐시 시스템으로 인해 칩 전체의 전력 소모량도 더불어 늘어나게 되었다[2].

최근 들어서는 내장형 기기와 이동 기기들이 널리 사용되고 있다. 이로 인해 마이크로프로세서에 대한 연구는 해당 기기의 성능 향상에 대한 연구뿐만 아니라, 시스템의 동작 주파수 및 복잡성 증가로 인한 온도문제 그리고 제한적인 유한 전지의 용량으로 인한 에너지 효율성 문제를 부각 시켰다. 내장형 시스템의 설계에서 캐시나 변환색인버퍼(TLB, Translation Lookaside Buffer)와 같은 메모리 시스템들은 전체 프로세서 에너지의 거의 절반 이상을 소비하고 아울러 프로세서 성능에도 직접적인 영향을 끼친다. 비록 정적전력(Static Power)도 프로세서 전력 소비에 많은 부분을 차지하고 그에 대한 연구도 많이 이뤄지고 있지만, 여전히 동적전력(Dynamic Power)은 저전력 설계에 있어서 중요한 부분이다[3]. 이 논문에서는 동적전력 소모를 줄일 수 있는 방법에 대해 논의한다.

한편, 내장형 시스템에서 실행되는 응용프로그램들은 실행 패턴의 높은 예측성과 동일한 메모리 접근 패턴을 가지는 지역성(Locality) 등으로 특징지을 수 있다. 이러한 특징은 그 시스템의 에너지 소비나 성능을 최적화시킬 수 있는 추가적인 기회를 제공한다. 내장형 시스템에서의 응용 프로그램 수행은 몇 개의 지역들로 나눠지며, 각각의 지역들은 메모리 영역의 적은 부분을 접근하게 된다. 내장형 시스템에서의 지역성에 대한 특징은 얼마나 많은 지역성들이 응용 프로그램에 포함되어 있느냐와 관계없이 각 지역성은 좁은 주소 공간 범위에 분포되어 있고, 드물게 몇 십개 이상의 캐시 라인을 뛰어 넘는다는 사실이다. 이러한 사실은 단지 소수개의 비트로 이루어진 주소 정보의 사용으로도 캐시로의 접근이 충분하다는 것을 의미한다. 관련 연구에서는 PowerStone 벤치마크 프로그램에서 사용되는 주소 비트 수의 변화에 따른 캐시의 적중률을 보여준다[4]. 해당 연구에서는 단지 5개 안팎의 태그 비트를 비교하는 것만으로도 충분한 캐시 적중률을 얻을 수 있다는 것을 보였으며, 극단적으로 0개의 태그 비트를 사용한다 하더라도 2% 이하의 비교적 적은 성능 감소만이 야기된다는 것을 알 수 있다.

본 논문에서는 이상의 사실을 바탕으로, 내장형 시스템에서의 응용 프로그램들은 메모리를 참조할 때 높은 지역성에 의한 좁은 범위의 메모리 접근 분포 특성을 가진다는 사실과 단지 몇 개의 지역성이 자주 접근된다는 특성을 이용해 캐시에서 특히 태그 부분의 에너지 소비를 줄일 수 있는 방법을 제안한다. 우선, 본 논문에서는 기존의 부분태그 관련 연구를 살펴보고, 해당 기법의 특징과 개선할 점을 알아본다. 이 분석을 바탕으로 기존 방식보다 효율적인 캐시 태그 압축을 사용하는 저전력 캐시 시스템을 설계하는 방법에 대해서 소개하고, 개선된 저전력 캐시의 모의실험을 통해 에너지 효율의 향상과 성능의 변화 정도를 알아본다.

www.kci.go.kr

II. 관련 연구

메모리 모듈의 에너지 효율을 높이기 위한 기술들이 다수 제안되어 왔다[5][6]. 이는 성능상의 필요 요소이기도 하며, 내장형 시스템에서는 작은 메모리의 사용이 에너지 효율을 증가시키기 위한 필수 요소이기 때문이다[7][8][9]. 한편, 오늘날의 프로세서들은 캐시 접근 실패율을 줄이기 위해 집합 연관 캐시들을 많이 사용한다. 하지만 많은 웨이(Way)를 가진 캐시는 적중률을 높일 수 있겠지만 필요하지 않은 캐시를 접근하기 때문에 에너지를 낭비하게 된다. 최근의 연구는 이런 문제점을 해결하기 위해 부분태그를 사용한다. 즉, 부분태그를 사용해 원하는 웨이를 선택적으로 접근하여 성능의 감소 없이 에너지 낭비를 줄일 수 있는 방법을 제시하고 있다.

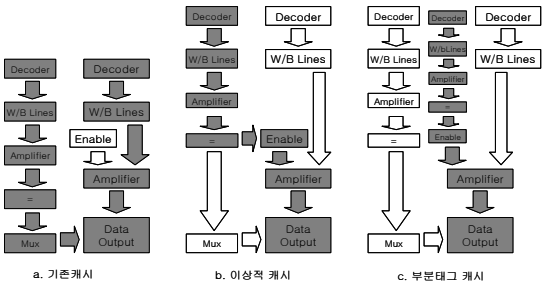

그림 1. 캐시 접근 경로
Fig. 1. Cache Access Path

그림 1은 캐시 접근 경로를 나타낸다. 왼쪽의 경로는 태그 어레이의 경로이며, 오른쪽의 경로는 데이터 어레이의 경로이다. 암영이 넣어진 박스는 활성화됨을 나타내는데 (a)는 기존 캐시를 나타낸다. 기존 캐시의 경우 태그와 데이터 라인을 동시에 접근하기 때문에 많은 에너지를 소비하게 된다. (b)는 이상적인 캐시를 나타낸다. 이는 전체태그를 사용하되 모든 웨이 중 하나의 웨이만 선택되어 활성화된다. 이는 최대의 에너지 효율을 얻을 수 있다. 하지만 태그 어레이의 비교기의 결과를 데이터 어레이의 Sense Amplifier로 전달하는데 긴 지연이 발생하여 접근시간이 10~20% 길어지게 된다. 그리고 이는 캐시의 크기가 커질수록 더욱 증가하게 된다. 따라서 이 방식은 에너지 효율을 얻을 수 있으나 성능 감소를 피할 수 없다. (c)는 부분태그를 사용하여 하나의 웨이를 활성화 시키는 동작이다. 부분태그는 작기 때문에 접근시간에 큰 영향을 끼치지 않으며, 이같이 구성하면 뚜렷한 성능의 저하 없이 전력소모량을 감소시킬 수 있다.

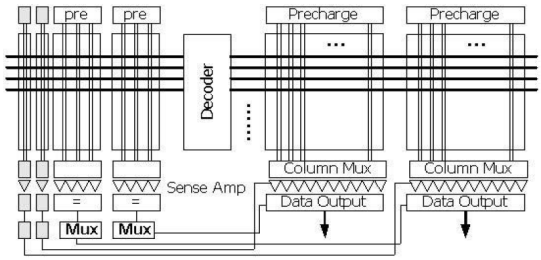

그림 2. 부분태그 캐시 구조
Fig. 2. Partial Tag Cache Structure

그림 2는 부분태그 캐시 구조이다. 이 방식은 매우 작은 부분태그를 추가하고 이는 기존의 태그처럼 접근된다. 각 웨이마다 이 최하위 비트들을 비교하게 되고 부합되는 비트 라인들만 활성화되게 된다. 이러한 구조를 활용하여 16KB 4-Way 집합연관캐시에서의 1차 레벨 캐시 전력소모량은 부분태그로 3-bit를 썼을 때 0.13μm과 0.35μm에서 각각 전체 캐시 접근 전력의 약 48%와 36%를 절약할 수 있었다. 데이터 캐시의 경우도 유사한 소비 전력의 감소를 보였다.

부분태그 캐시의 에너지 효율성은 좋으나 대상이 집합 연관캐시에 한정된다. 그리고 주어진 방식의 가장 큰 문제는 '거짓 성공(False Hit)'이다. 거짓 성공이란 원래 명령어의 태그와 캐시에 저장된 태그의 값을 비교 시킬 때에는 사실상 전체 태그가 필요한데, 부분태그를 사용하게 됨으로 인해 실제로는 실패지만 성공이라고 판단되는 경우이다. 이는 성공이라고 가정하고 계속 진행되기 때문에 나중에 페널티가 심각해지게 된다. 본 논문에서는 부분태그를 쓰지만 다수의 지역성을 저장함으로써 '거짓 성공'을 최대한 줄이는 기법을 제시한다.

한편, 그림 3은 부분태그 캐시 방식을 이용하는 또 다른 연구이다. 이 방식은 TOB(Tag Overflow Buffer)라는 레지스터를 사용하여 태그의 상위 부분을 저장한다. 이는 마치 캐시 외부에 있는 레벨 0 캐시와 비슷하게 동작하며, 응용 프로그램의 지역성을 식별하게 된다. 이 같은 방식도 앞서 언급한 거짓 성공을 줄이는데 도움을 줄 수 있다. 동작 방식은 다음과 같다. 우선, TOB와 캐시는 동시에 접근되며, TOB Hit가 발생할 경우 감소된 태그 비트수를 사용하여 캐시를 접근하게 된다. 이 경우, 캐시의 Hit/Miss의 동작 방식은 기존의 방식과 동일하다. 즉, TOB 접근 성공이 발생하면 적은 비용으로 캐시 접근을 할 수 있다[10].

www.kci.go.kr

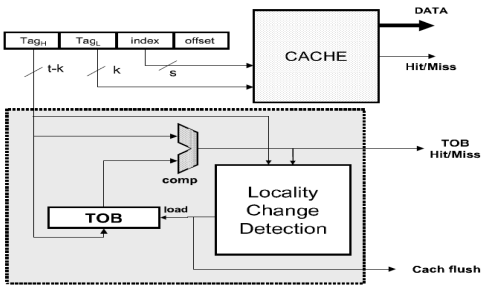

그림 3. 부분태그 직접사상 캐시
Fig. 3. Partial Tag Direct Mapped Cache (PTDMC)

표 1은 TOB를 활용한 부분태그 캐시의 에너지 소비 감소를 보여준다. 1KB 크기, 4Byte의 라인 크기를 가지는 부분태그 명령어 캐시에서 평균 51%에 해당하는 태그 영역의 소비 전력 감소를 가져왔으며, 이로 인해 전체 캐시 시스템에서 소비되는 전력은 약 21% 감소되었다. 최적의 태그 수는 프로그램마다 다르지만 0~3개의 비트만으로도 원하는 성능을 보여준다.

표 1. 부분태그 직접사상캐시의 에너지 소비
Table 1. Energy Consumption of PTDMC

Benchmark	k_{opt}	Tag Energy (pJ)			Cache Energy (pJ)		
		Original	Optimized	Δ	Original	Optimized	Δ
*adpcm	1	5,65e05	3,25e05	42,6%	1.37e6	9.21e5	32.8%
bcnt	2	9,76e3	5,43e3	44,4%	2.34e5	1.89e5	19.4%
blit	0	1,50e5	0	100%	3.58e5	2.81e5	21.5%
compress	3	1,05e6	6,08e5	41,9%	2.51e6	2.05e6	18.4%
crc	2	2,49e5	1,39e5	44,1%	5.95e5	4.79e5	19.4%
*des	1	9,69e5	5,60e5	42,3%	2.34e6	1.74e6	25.7%
engine	0	2,73e6	0	100%	6.54e6	5.15e6	21.3%
fir	2	1,17e5	6,50e4	44,4%	2.81e5	2.26e5	19.4%
g3fax	2	7,51e6	4,20e6	44,1%	1.80e7	1.45e7	19.4%
jpeg	3	3,08e7	1,79e7	41,8%	7.37e7	6.02e7	18.4%
pocsag	2	3,52e5	1,95e5	44,4%	8.45e5	6.80e5	19.5%
qurt	2	8,75e3	4,83e3	44,8%	2.11e4	1.70e4	19.5%
ucbqsort	2	1,70e6	9,41e5	44,5%	4.09e6	3.29e6	19.4%
v42	3	1,94e7	1,12e7	42,3%	4.68e7	3.81e7	18.5%

하지만 이 방식 역시 프로그램의 길이가 길어지고 지역성이 자주 바뀌는 프로그램이라면 매우 취약해질 수 있다. 즉, 거짓 성공이 자주 나타날 수 있다. 지역성이 바뀌게 되면 캐시의 모든 내용을 버려야 하기 때문에 페널티도 상당하다. 본 논문에서는 이런 문제점을 해결할 수 있는 기법을 알아본다.

III. 압축 태그 캐시 설계

메모리 계층은 작고 빠른 메모리를 프로세서 가까이에 위치시킴으로써 최상위 계층에서의 적중된 접근을 빠르게 처리할 수 있다. 적중률이 충분히 크면 메모리는 최상위 계층의 접근 속도와 비슷한 접근 속도를 갖고, 최하위 계층의 메모리 크기와 비슷한 크기를 가진 것처럼 작동한다. 상위 레벨 캐시에서는 이런 지역성이 높게 나타나며 이를 이용해 태그의 효율을 높이게 된다. 한편, 캐시 내의 태그는 캐시내의 워드가 프로세서가 요청한 워드인지를 식별하는 데 필요한 정보를 포함하는데 캐시의 색인(Index)으로 사용되고 난 나머지 비트들이 태그가 된다. 이 때 사용되지 않는 비트들이 많을수록 태그 비트들은 많아지는데 이는 에너지 측면에서 보았을 때 비효율적이다. 전체 태그 중에서 상위 태그들은 위에서 설명한 지역성을 표현하게 되는데 이런 특징을 이용해 전체 태그를 감소시켜 에너지 효율을 높이고 비슷한 성능을 나타낼 수 있는 저전력 캐시를 제안한다.

그림 4는 본 논문에서 제안된 기법을 활용하기 위한 수정된 명령어 세트 구조(ISA)를 나타낸다. n은 주소 비트, b는 오프셋 비트, s는 인덱스 비트, th와 tl은 각각 태그비트들의 상위, 하위비트들의 크기를 나타낸다. 암영이 넣어진 th는 위에서 설명한 응용 프로그램의 지역성을 나타내게 된다. 지역성이 높게 나타날수록 th는 길어지고 tl은 짧아지게 된다. 태그 압축 캐시에서는 이 tl만 사용하게 되고 비트수가 줄어든 만큼 에너지를 절약할 수 있다.

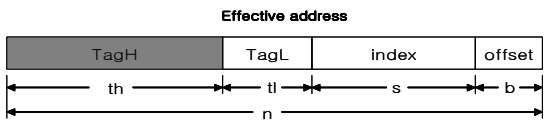

그림 4. 명령어 세트 구조
Fig. 4. Instruction Set Architecture

나머지 th는 별도의 레지스터에 저장되는데, 본 논문에서는 이 공간을 지역버퍼(Locality Buffer)라 명명 한다. 그림 5에 제시되어 있는 지역버퍼는 몇 개의 라인을 가지고 있고, 그 개수에 따라 지역압축 비트(Locality Compressed Bit)가 결정된다. 예를 들면 지역 버퍼의 엔트리 개수가 4라면, 지역압축 비트는 00, 01, 10, 11 두 비트로 표현될 수 있다. 이로써 비교되는 태그는 지역 버퍼에 쓰이는 비트를 제외한 나머지 하위 비트들과 지역 압축 비트들을 더한 것이 되고, 태그를 적게 쓰기 때문에 그만큼의 에너지 소모를 줄일 수 있게 된다.

지역버퍼의 엔트리 개수는 소프트웨어의 개발단계에서 결정되는데 지역버퍼의 엔트리 개수를 많게 한다면 적중실패를 줄일 수 있겠지만 기존의 캐시에 추가되는 하드웨어의 증가를 의미할 뿐만 아니라 지역압축 비트의 증가도 발생시키기 때문에 전력감소의 효과가 떨어지게 된다. 따라서 프로그램에 따

www.kci.go.kr

라 적절한 엔트리의 개수를 고려해 보아야 하며, 지역성이 높아질수록 적은 개수로도 충분하다.

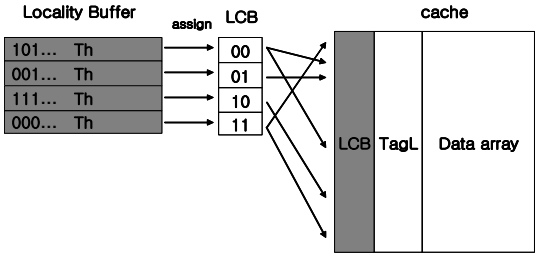

그림 5. 지역 버퍼
Fig. 5. Local Buffer

그림 6은 지역버퍼를 사용한 태그 압축 캐시의 기본구조도이다. 먼저 색인으로 캐시를 접근하게 된다. 이 때 태그 비교는 하위 부분태그로 행해지며 접근 실패가 되면 기존 캐시의 실패와 동일하게 동작한다. 이후 외부 메모리에서 올바른 값이 오면 해당라인을 갱신한다. 하위태그 비교가 적중되었다면 해당라인의 지역압축 비트를 보고 지역버퍼를 접근하게 된다. 그 다음, 지역버퍼에 저장된 값과 상위태그를 비교해서 성공여부를 가린다. 성공되었다면 비로소 접근했던 캐시의 값은 유효하게 되고 캐시접근은 끝나게 된다. 반대로 실패가 되면 나머지 지역 버퍼에 같은 값이 있는지 살펴보게 된다. 같은 값을 찾게 되면 캐시의 값은 유효하며 해당 캐시라인의 지역압축 비트만 바꿔주면 된다. 문제가 되는 것은 나머지 지역버퍼에도 원하는 값이 없었을 때이다. 이 경우 교체(replacement) 메커니즘이 필요하며, 교체 과정은 다음과 같이 진행 된다.

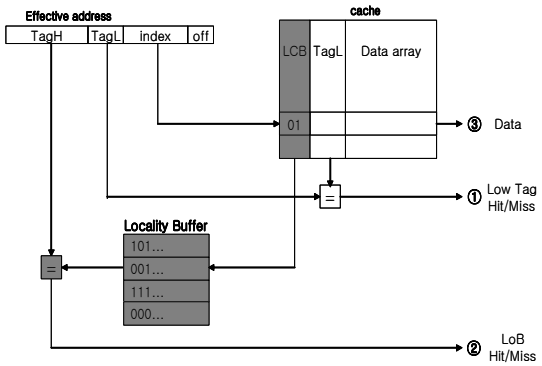

그림 6. 지역 버퍼를 사용하는 저전력 캐시 구조
Fig. 6. Low Power Cache using Local Buffer

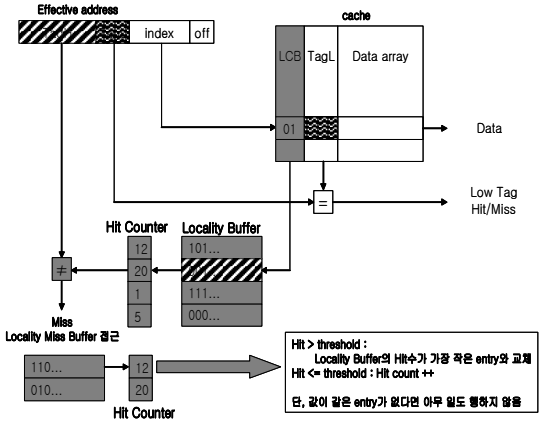

그림 7. 지역 버퍼 교체 메커니즘
Fig. 7. Local Buffer Replacement Mechanism

앞서 설명한 대로 캐시의 교체는 모든 지역버퍼에 해당 지역성이 없을 때, 즉 지역 버퍼에 내용이 바뀌어졌을 때 행해진다. 이런 경우에 단순히 캐시의 지역압축 비트를 보고 갱신할 수 있지만 이는 모든 라인을 살펴봐야 하기 때문에 오버헤드가 너무 크다. 따라서 본 논문에서는 새로운 하드웨어를 추가해 오버헤드를 최소화시킬 수 있는 방법을 제안한다. 우선, 이를 위하여 추가된 하드웨어는 다음과 같다.

- 지역실패 버퍼 (Locality Miss Buffer) : 지역버퍼의 엔트리 값을 바꾸기 위한 하드웨어이다. 지역버퍼에 없는 지역성들을 저장하게 되며 각 엔트리마다 계수기를 갖는다.
- 적중 계수기 (Hit Counter): 엔트리가 적중되면 값을 증가시킨다. 지역버퍼와 지역실패 버퍼 엔트리마다 하나씩 추가.

교체과정은 다음과 같다. 먼저 캐시에서의 교체과정은 기존 캐시에서 쓰이던 방식, 가령 LRU(Least Recently Used), 등을 사용하며 2차 레벨 캐시에서 값이 올라와 갱신되면서 올바른 지역압축 비트를 저장하고 계수기의 값도 증가시킨다. 지역버퍼의 계수기는 적중될 때마다 증가하며, 지역버퍼에서 실패가 생기면 지역실패버퍼를 접근하게 된다. 지역실패버퍼에 빈 엔트리가 있다면 해당 태그를 저장하게 되며 지역실패버퍼에 같은 값이 있으면 해당 계수기를 증가시키고 한계값(threshold)을 초과하는지 살펴본다. 넘지 않는다면 단순히 계수기의 값만 증가시키게 되고 2차 레벨캐시에서 올라온 값은 바로 CPU에 전달된다. 만약 한계값을 넘는다면 이때 지역버퍼의 교체과정이 시작된다. 지역버퍼의 엔트리 중

www.kci.go.kr

에 제일 작은 적중수를 갖는 엔트리와 교체되는데, 이 값도 한계값을 넘으면 한계값을 조정한다. 적중수가 제일 적은 엔트리와 교체된다면 계수기 값은 리셋('0')되며, 해당캐시 라인의 데이터 값도 2차 레벨에서 올라온 값으로 대체된다. 이상의 내용이 그림 7에 제시되어 있으며, 표 2을 통해 전체적인 동작 방식이 요약되어 있다.

표 2. 적중 성공 및 실패에 따른 캐시 동작
Table 2. Cache Operation based on Hit/Miss

Low Tag (Cache)	High Tag (Locality Buffer)	Locality Miss Buffer	동작
Hit	Hit	-	기존캐시에서의 적중과 동일. Hit counter ↑.
	Miss	Hit	Hit counter ↑. thd값을 초과하면 지역버퍼 교체. 아니면 외부메모리에서 CPU 전달.
		Miss	비어있다면 추가. 아니면 아무 일도 행하지 않음. 외부메모리에서 CPU 전달.
Miss	Hit	-	기존캐시에서의 실패와 동일. 해당라인 갱신(LRU).
	Miss	Hit	Hit counter ↑. thd값을 초과하면 지역버퍼 교체. 아니면 외부메모리에서 CPU 전달.
		Miss	비어있다면 추가. 아니면 아무 일도 행하지 않음. 외부메모리에서 CPU전달.

IV. 모의실험 및 성능 분석

본 논문의 제안을 검증하기 위해서, 슈퍼스칼라 프로세서의 사이클 수준 시뮬레이터인 SimpleScalar ARM 버전을 사용한다. 이는 기존의 SimpleScalar 시뮬레이터에 ARM 바이너리로 컴파일된 프로그램을 실행하기 위해 ARM ISA의 지원이 가능하도록 개조되었다[11]. 전력소모량의 측정을 위해서 CACTI를 사용하였으며, 0.18um 기술을 가정하였다[12]. 모의실험을 위한 환경 인자는 표 3과 같다. 본 본 논문에서의 벤치마크 프로그램은 EEMBC를 기반으로 만든 MiBench[13]를 사용한다. 본 논문에서는 그 가운데 가장 활발하게 사용되는 응용 분야인 멀티미디어와 통신 영역의 응용 프로그램들을 선택하여 모의실험을 수행 하였다.

표 3. 모의실험 인자
Table 3. Simulation Parameter

인자 종류		인자 값
정수 수치 연산기(IALU)		1
부동 소수점 수치 연산기(FPALU)		1
정수 곱셈/나눗셈 연산기(IMUL)		1
부동 소수점 곱셈/나눗셈		1
1 사이클 당 명령어 인출 속도		1
RUU(Register Update Unit) 크기		8
LSQ(Load Store Queue) 크기		8
분기 예측기의 종류		Bimodal
분기 예측기의 테이블 크기		128
분기 예측 실패 손실(branch penalty)		3
BTB의 엔트리 수/집합 연관도		128/직접사상
레벨 1 캐시	크기	1KB ~ 8KB
	집합 연관	직접사상
	블럭 크기	32B
	접근 성공 지연 사이클	1 사이클
메모리 접근 성공 지연 사이클		32 사이클

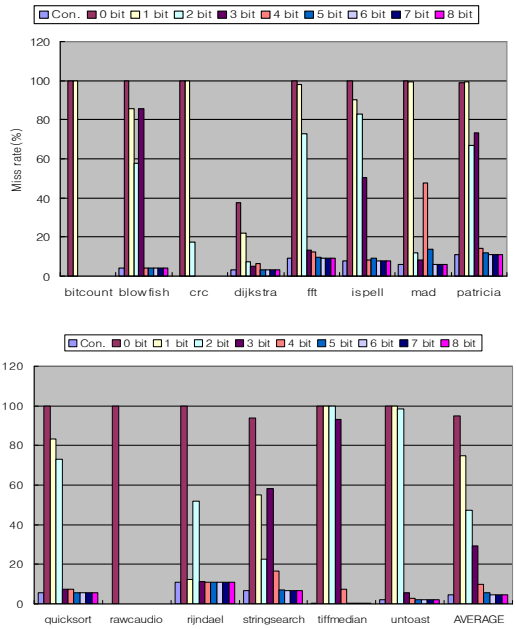

그림 8. 부분태그 비트수에 따른 실패율
Fig. 8. Miss Ratio vs. the Number of Partial Tag Bits

그림 8은 부분태그의 비트수에 따른 캐시 접근 실패율을 비교한 것이다. 4KB의 캐시, 4개의 지역버퍼, 2개의 지역미스버퍼, 1 threshold를 사용한 결과이다. Con.는 기존의 캐시에서의 실험 결과이며, 그림 8에서는 0~8 비트를 사용했을 때의 캐시 접근 실패율을 보인다. 그림 8에서, 대체적으로

www.kci.go.kr

1~6 비트만을 사용하였을 때, 전체 태그를 사용할 때의 실패율과 완전히 같아짐을 알 수 있다. 평균적으로 5비트를 사용할 때부터 실패율은 1% 내의 차이를 보이며 6비트를 사용할 때 완전히 같아진다. 성능감소가 허용된다면, 에너지 소모를 더 줄이기 위해 사용 비트수를 더욱 감소시킬 수 있다.

그림 9는 지역버퍼의 엔트리 수(LoB)에 따른 실패율을 나타낸다. 지역 버퍼의 엔트리 수가 늘어나면 그 자체로 하드웨어적 오버헤드가 될 뿐만 아니라 그만큼 캐시라인마다 저장되는 지역압축 비트의 수도 늘어나게 된다. 이럴 경우, 성능저하는 없더라도 전력소비의 효율성은 떨어지게 된다. 따라서 적절한 지역버퍼 엔트리 수의 결정은 중요하다. 그림 9에서 나타나듯이 지역버퍼의 개수가 2개일 때는 높은 실패율을 보인다. 하지만, 지역버퍼의 엔트리 수가 4개일 때는 기존 캐시에 비해 약 0.8%의 차이를 보이며 8개일 때는 약 0.08%의 차이를 보인다. 즉 4개 이상의 지역버퍼를 두는 것이 좋으며, 실험 결과 적절한 지역버퍼의 엔트리 수는 4개로 보인다.

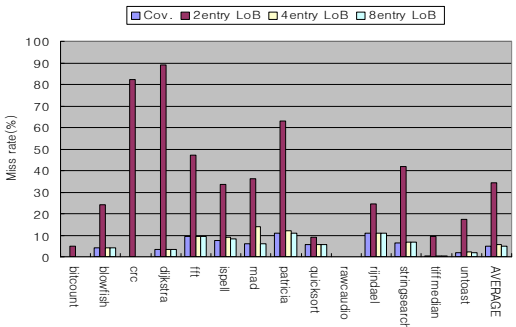

그림 9. 지역버퍼 엔트리수에 따른 실패율
Fig. 9. Miss Ratio vs. the Number of Local Buffer Entries

지역 실패 버퍼의 threshold는 지역버퍼의 엔트리를 교체할 때 필요한 값이다. 지역실패버퍼 엔트리가 threshold를 넘게 되었을 때 지역버퍼로 들어가게 된다. 그림 10에서 보듯이 threshold에 따른 실패율은 프로그램마다 다른 특성을 보인다. threshold에 관계없이 같은 실패율을 보이는 프로그램은 blowfish, crc, quicksort, rawcaudio, rijndael, tiffmedian, untoast이며, threshold가 커질수록 실패율이 커지는 프로그램은 bitcount, dijkstra, fft, patricia, stringsearch이다. 한편, threshold에 관계없이 다른 실패율을 보이는 프로그램은 ispell, mad이다. 일반적으로 내장형 시스템은 전용 시스템으로 사용되어 특정 응용 프로그램만을 수행하는 경향이 있다. 따라서 이를 반영하여 응용 프로그램의 특성에 따라 달라질 수 있는 threshold 값을 인식하고, 이를 반영하여 시스템 개발 초기에 미리 그 값을 결정하는 것이 성능상 유리하다. 본 논문에서는 대체적으로 threshold가 커질수록 실패율이 커지기 때문에 한번 지역 실패 버퍼에 등록되고 한번 더 접근되면 바로 지역버퍼에 올라가게 되는 값인 '1'을 사용한다.

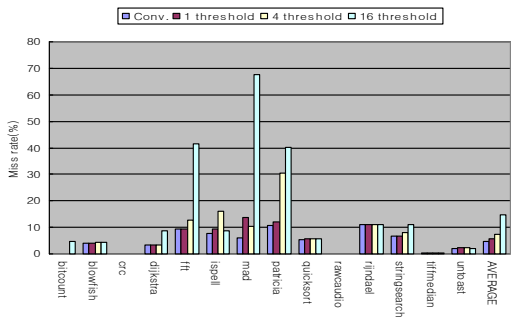

그림 10. Threshold 값에 따른 실패율
Fig. 10. Miss Ratio vs. Threshold Value

그림 11은 지역 실패 버퍼의 크기에 따른 실패율이다. 크기에 관계없이 같은 실패율을 보이는 프로그램은 blowfish, crc, quicksort, fft, quicksort, rawcaudio, rijndael, stringsearch이며, 크기가 커질수록 실패율이 작아지는 프로그램은 bitcount, dijkstra, ispell, mad, tiffmedian, untoast이다. 하지만 대부분 2개 이상을 사용했을 때, 전체 태그 사용 시의 실패율과 같아지게 된다. 특히 bitcount, patrica, tiffmedian, untoast의 경우 크기에 따라 많은 차이를 보이므로, 지역실패버퍼의 크기 역시 초기 개발 시에 고려되어야 한다.

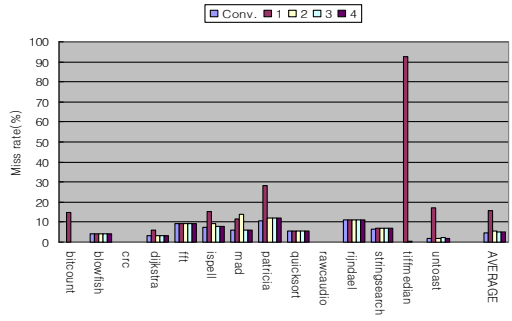

그림 11. 지역실패버퍼의 크기에 따른 실패율
Fig. 11. Miss Ratio vs. the Number of LMB Entries

www.kci.go.kr

표 4. 에너지 소모량 비교
Table 4. Energy Consumption

Benchmark	Opt. Tag Length	Cache Energy(pJ)		
		Full Tag	Partial Tag	▲
bitcount	2	3.95E+05	3.20E+05	**19.2%**
blowfish	4	4.08E+05	3.40E+05	**16.4%**
crc	3	4.80E+05	3.95E+05	**17.2%**
dijkstra	5	5.23E+05	4.40E+05	**15.4%**
fft	5	4.85E+05	4.08E+05	**15.9%**
ispell	5	6.98E+05	5.85E+05	**16.0%**
mad	6	2.07E+05	1.73E+05	**16.2%**
patricia	6	9.73E+05	8.23E+05	**15.4%**
quicksort	5	3.68E+05	3.08E+05	**16.4%**
rawcaudio	1	2.93E+05	2.11E+05	**27.8%**
rijndael	3	2.63E+05	2.20E+05	**16.2%**
str search	5	1.48E+03	1.24E+03	**15.9%**
tiffmedian	5	1.15E+06	8.73E+05	**24.3%**
untoast	5	1.95E+05	1.57E+05	**19.5%**
average				**18.0%**

표 4는 전체태그를 사용했을 때와 부분태그를 사용했을 때의 전체 캐시에서 소모되는 전력량을 나타낸다. 최적의 비트 수는 기존 전체태그를 사용했을 때의 실패율과 비교해 0.5% 이내의 차이를 보이는 것으로 정하였다. 전체적으로 본 모의실험을 통하여 평균적으로는 18%의 에너지 소모를 줄일 수 있었다.

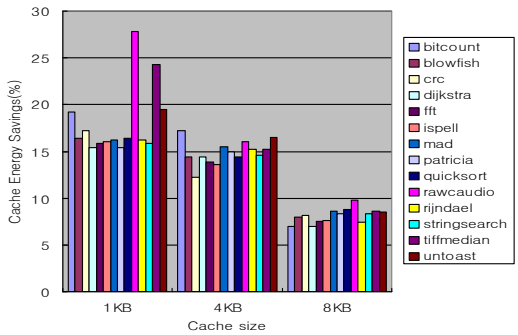

그림 12. 캐시의 크기에 따른 전력소모량
Fig. 12. Energy Consumption vs. Cache Size

그림 12는 캐시의 크기에 따른 전력소모량이다. 제안하는 캐시 구조는 작은 크기의 캐시가 사용되는 시스템에 더욱 효율적인 방식이라 할 수 있다. 캐시의 크기가 1KB일 때 최대 27%, 4KB와 8KB의 캐시에서는 각각 14%와 9%의 전력량을 줄일 수 있었다. 한편, 본 논문의 제안으로 인해 추가적으로 필요한 메모리 요소는 4KB~16KB 전체 캐시 크기 기준으로 0.08%~0.30%에 해당하는 매우 작은 수치이다. 따라서 본 논문의 제안으로 인해 추가적으로 요구되는 메모리 용량과 그로 인해 요구되는 추가 전력소비량은 사실상 무시할 만한 수준이라 판단된다.

V. 결 론

본 논문에서는 캐시 시스템의 소비 전력 감소를 위한 연구 가운데 하나로, 캐시 태그 압축을 통한 소비 전력 감소 방안을 제안하였다. 높은 지역성을 보이는 내장형 프로그램들의 특성을 이용하면 캐시에서 많은 부분을 차지하는 태그의 비트 수를 줄여 성능의 저하 없이 에너지 소모를 줄일 수 있게 된다. 제안된 기법은 태그는 줄어드는 대신 지역버퍼와 지역실패버퍼, 지역 압축비트를 필요로 하게 된다. 하지만 이들은 간단한 하드웨어로 구현되기에 큰 오버헤드가 되지 않는다. 모의실험을 통해서 검증한 결과, 4개의 지역버퍼와 2개의 지역 미스버퍼의 추가로 전체 태그 주소 공간의 약 1/4 정도만 사용할 수 있게 되었으며, 그 결과 전체 태그를 사용했을 때보다 성능의 감소 없이 최대 27%, 평균 18%의 캐시 에너지를 줄일 수 있었다. 아울러 본 논문에서 제안된 방식은, 캐시의 용량이 작을 때 보다 더 우수하다. 따라서 캐시의 용량이 제한되는 소규모 내장형 시스템에서 보다 더 유용하게 쓰일 수 있다. 추후 연구를 통해 보다 더 효과적인 알고리즘과 적은 용량의 메모리를 요구하는 방안을 연구하여 향후 내장형 시스템의 변화에 대응할 필요가 있다고 판단된다.

참고문헌

[1] Patterson, D. A., and Hennessy, J. L. "Computer architecture: a quantitative approach" (Morgan Kaufman, 2007, 4th Ed.)
[2] S. Segars, "Low power design techniques for microprocessor" Tutorial, International solid-State Circuit conference, Feb.2001
[3] ARMv7-M Architecture Reference Model, ARM DDI 0403C, 2010.
[4] A. Malik, B. Moyer, D. Cermak, "A Lower Power Unified Cache Architecture Providing Power and Performance Flexibility," ISLPED'00, 2000, pp. 241~243

www.kci.go.kr

[5] Gennady Pekhimenko, et al., "Linearly compressed pages: a main memory compression framework with low complexity and low latency", PACT '12 pp. 489~490, 2012

[6] H. Kim, C. Rhee, and H. Lee, "Address generation for lossless frame memory compression in an H.264/AVC encoder", Proceedings on ICCE, pp. 570~571, 2013

[7] Xi Chen, Lei Yang and Li Shang, "C-Pack: A High-Performance Microprocessor Cache Compression Algorithm", IEEE Transactions on VLSI Systems, pp. 1196~1208, 2010

[8] Hadi Hajimiri and Kamran Rahmani, "Compression-aware dynamic cache reconfiguration for embedded systems", IEEE IGCC 2011, pp. 71-80, June 2012,

[9] Mathew, S. and Jagadeeswari, M., "Low power L2 cache design using partially tagged bloom filter and hotline check", International Conference on ICCCI, pp. 1~6, Jan. 2013

[10] M. Loghi, P. Azzoni, M. Poncino, Tag overflow buffering: an energy-efficient cache architecture, in: Proceedings of the Design, Automation and Test, pp. 520~525, March 2005.

[11] D. Burger, A. Kagi, and M. Hrishikesh. "Memory hierarchy extensions to SimpleScalar 3.0", Technical Report TR99-25, University of Texas at Austin, April 1999.

[12] Naveen Muralimanohar, Rajeev Balasubramonian, Norman P. Jouppi, "CACTI 6.0: A Tool to Model Large Caches", HPL-2009-85, Tech. Report, Compaq Western Research Lab, 2009.

[13] Guthaus, M. R. et al., "MiBench: A free, commercially representative embedded benchmark suite", IEEE International Workshop on Workload Characterization, pp. 3~14, Dec. 2001

저 자 소 개

곽 종 욱

1998: 경북대학교
컴퓨터공학과 공학사.

2002: 서울대학교
컴퓨터공학과 공학석사.

2006: 서울대학교
전기컴퓨터공학부 공학박사

현 재: 영남대학교
컴퓨터공학과 부교수

관심분야: 컴퓨터구조,
임베디드 시스템,
고성능 컴퓨팅

Email : kwak@yu.ac.kr

www.kci.go.kr