내장형 시스템을 위한 저전력 2-레벨 캐쉬 메모리의 설계

이종민*, 김순태*, 김경아**, 박수호***, 김용호***
*한국정보통신대학교 임베디드시스템 연구실
**KT 미래기술연구소
*** KT 인프라연구소
e-mail : ihooda@icu.ac.kr

Low-Power 2-level Cache Architectures for Embedded System

Jong-Min Lee*, Soon-Tae Kim*, Kyung-Ah Kim**, Su-Ho Park***, Yong-Ho Kim***
*Embedded Computing Lab, Information and Communication University
**Service Development Laboratory, KT
***Network Management & Support Center, KT

요 약

온칩(on-chip) 캐쉬는 외부 메모리로의 접근을 감소시키는 중요한 역할을 한다. 본 연구에서는 내장형 시스템에 맞추어 설계된 2-레벨 캐쉬 메모리 구조를 제안하고자 한다. 레벨1(L1) 캐쉬의 구성으로 작은 크기, 직접사상(direct-mapped) 그리고 바로쓰기(write-through)를 채용한다. 대조적으로 레벨2(L2) 캐쉬는 일반적인 캐쉬 크기와 집합연관(set-associativity) 그리고 나중쓰기(write-back) 정책을 채용한다. 결과적으로 L1 캐쉬는 한 사이클 이내에 접근될 수 있고 L2 캐쉬는 전체 캐쉬의 미스율(global miss rate)을 낮추는데 효과적이다. 두 캐쉬 계층간 바로쓰기(write-through) 정책에서 오는 빈번한 L2 캐쉬 접근으로 인한 에너지 소비를 줄이기 위해 본 연구에서는 One-way 접근 기법을 제안 하였다. 본 연구에서 제안한 2-레벨 캐쉬 메모리 구조는 평균적으로 26%의 성능향상과 43%의 에너지 소비 그리고 77%의 에너지-지연 곱에서 이득을 보여주었다.

1. 서 론

중앙처리장치(CPU)와 메모리(Main memory)간의 성능 격차는 더욱 벌어지고 있다. 비록 캐쉬 메모리가 중앙처리장치와 메모리간 성능차이를 완화하기 위해 사용되고 있지만 매년 빠르게 발전하는 중앙처리장치와 더디게 발전하는 메모리와의 속도 간격은 더욱 커지고 있다. 게다가 비디오 스트리밍(video-streaming), 이미지 프로세싱(image-processing), 고성능 무선통신(high speed wireless communication) 등과 같은 프로그램의 등장은 내장형 시스템에 대한 성능향상과 저전력 소비에 대한 요구를 더욱 높이게 되었다. 이러한 요구를 충족하기 위해 새로운 메모리 구조를 탐험하는 것은 매우 중요한 일이다. 왜냐하면, 메모리 시스템은 시스템 전력의 많은 부분을 소비하고 성능에 직접적으로 영향을 끼치는 부분이기 때문이다. 메모리 시스템의 구성원들 중 캐쉬는 매우 중요한 부분이다. 내장형 프로세서에서 대략 30% 이상의 전력이 캐쉬에 의해 소비된다는 관련 연구가 있다[2].

일반적으로 내장형 프로세서는 낮은 클럭 주파수를 가지고 있기 때문에 캐쉬의 긴 접근 시간이 중요한 문제가 아니었다. 따라서 <표1>에 보인 바와 같이 내장형 시스템에서는 일반적인 크기의 집합연관 1-레벨 캐쉬가 빈번하게 사용되고 있다. 그러나 최근의 높은 클럭 주파수의 내장형 프로세서에서는, 이러한 긴 접근 시간은 많은 사이클을 필요로 한다. 또한, 집합연관사상(set-associativity) 캐쉬는 많은 전력을 소비함에도 불구하고 고성능을 위해서 내장형 시스템에서 채용되어왔다. 또 다른 문제로 어플리케이션에 따라 코드 부분과 데이터 부분의 워킹셋(Working-set) 크기가 다르기 때문에 대다수의 내장형 시스템에서 채용된 하바드(Harvard) 캐쉬 구조는 캐쉬의 용량 활용 측면에서 유연하지 않다.

<표 1> 내장형 프로세서들의 캐쉬 구성

Arm920T	Arm1020E	3rd gen Xscale	PowerPC405
16K/16K 32-way	32K/32K 64-way	32K/32K 4-way	16K/16K 2-way

위의 관찰을 토대로 본 연구에서는 고성능 프로세서에서 통용되는 2-레벨 캐쉬 구조와 유사하지만 자원 제한적인 내장형 시스템에 적합한 2-레벨 캐쉬 구조를 제안한다. 제안된 캐쉬 구조는 작은 크기이며 직접사상(direct-mapped)의 L1 캐쉬와 일반적인 크기이며 집합연관사상의 L2 캐쉬로 구성된다. 작은 크기와 직접사상을 채용한 L1 캐쉬는 저전력 소비와 빠른 접근 시간을 보장 할 수 있으며[6] 나중쓰기 정책과 집합연관사상의 L2 캐쉬는 L1 캐쉬에서의 미스를 보완하기 위해서 채용되었다. L1 캐쉬에서 L2 캐쉬로의 쓰기동작으로 인한 사이클 지연을 줄이기 위해서 쓰기버퍼(write-buffer)가 두 캐쉬 사이에 위치하고 있다. 그리고 쓰기버퍼에서 L2 캐쉬로 쓰기 연산시의 전력 소비를 줄이기 위해 One-way 접근(one-way access)이라고 명명된 기법을 제안 하였다.

아래의 내용은 다음과 같다. 2장에서 관련된 연구에 대해서 논의하고 3장에서 제안된 2-레벨 캐쉬 구조에 관해 설명한다. 4장에서 실험 준비와 실험결과에 대한 분석을 하고 마지막으로 5장에서 결론을 맺는다.

2. 관련 연구

지금까지 다양한 종류의 캐쉬 구조들이 저전력 소비와 성능향상을 위해 제안되었다. 그 중 제안된 캐쉬 구조와 관련 있는 연구들은 다음과 같다.

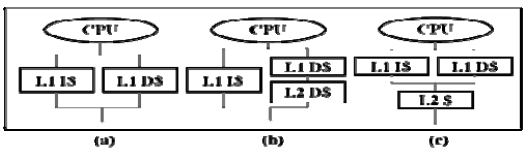
중앙처리장치와 명령어 캐쉬 사이에 위치한 작은 크기의 필터 캐쉬(Filter cache) [1, 2, 3]가 있다. 필터 캐쉬가 작은 크기이기 때문에 명령어가 필터 캐쉬에서 히트한다면 평균적인 캐쉬 접근 시간과 전력 소비를 줄일 수 있다. 그러나 작은 크기의 필터 캐쉬는 일반적으로 히트율이 낮기 때문에 오히려 캐쉬 접근 시간을 증가시킬 수도 있다. 이러한 문제를 해결하기 위해 언제 필터캐쉬를 접근하지 않고 바로 명령어 캐쉬로 접근할 것인지 결정하는 예측기반의 필터 캐쉬[2, 3]가 제안 되었다. 또 다른 종류의 캐쉬 구조로 작은 사이즈(1K-2K)이며 데이터 캐쉬와 수평적으로 위치하는 미니 데이터 캐쉬(Mini-data cache)[4]가 있다. 데이터 캐쉬나 미니 데이터 캐쉬 어느 곳에 데이터를 위치시킬지 결정하기 위해 어플리케이션 데이터들의 특성을 분석한 후 이를 적용하기 위해 컴파일러의 도움이 필요하다. 위의 언급된 캐쉬 구조들은 명령어 혹은 데이터 캐쉬를 목표로 하였던 반면, 본 연구에서는 명령어와 데이터 캐쉬를 동시에 목표로 하였다.

3. 제안된 캐쉬 구조

3.1 기본 구조

본 연구에서 사용된 기본 프로세서 구조는 내장형 시스템을 타겟으로한 인텔 XScale[9]이다. 인텔 XScale은 7-8단계의 파이프라인 구조이며 single-issue 수퍼파이프라인 코어와 32KB 32-웨이의 집합연관 명령어/데이터 캐쉬구조를 가지고 있다. 최근의 3세대 XScale 코어는 본 연구에서 기본구조로 채택한 4-웨이 명령어/데이터 캐쉬를 가지고 있다.

3.2 제안된 2-레벨 캐쉬 구조



(그림 1) 일반적인 1-레벨 캐쉬 구조(a)와 제안된 2-레벨 캐쉬 구조 (b)(c)

그림 1(a)는 내장형 시스템에 많이 채용된 집합 연관사상 하바드(Harvard) 캐쉬 구조를 보여준다. 일반적인 1-레벨 집합연관사상 하바드 캐쉬 구조는 3가지의 단점을 가지고 있다. 첫째로, 집합연관 캐쉬는 긴 읽기 시간을 유발한다. 이것은 주로 모든 캐쉬 웨이를 읽은 후에 태그(tag)를 비교하고 마지막으로 요청된 한 웨이를 선택하기 때문이다. 예를 들어, XScale은 캐쉬 읽기 연산을 위해 2사이클을 요구한다. 두 번째로, 집합연관 캐쉬는 히트시간을 줄이기 위해 다수의 캐쉬 웨이가 접근되어야 하기 때문에 많은 전력을 소비한다. 세 번째로, 일반적인 하바드 캐쉬 구조는 어플리케이션의 코드와 데이터 워킹셋 크기에 따라 캐쉬 용량을 효율적으로 이용할 수 없다.

<표 2> CACTI[7]로 추출한 다양한 캐쉬 구조의 접근시간 (pS), 에너지 소비 (pJ), One-way 접근 에너지 소비 (pJ)

configuration	1K 1W	2K 1W	8K 4W	16K 4W	32K 4W	28K 7W
Latency	512	522	992	1033	1080	1127
Power	320	330	904	939	995	1693
1W-power			421	467	505	764

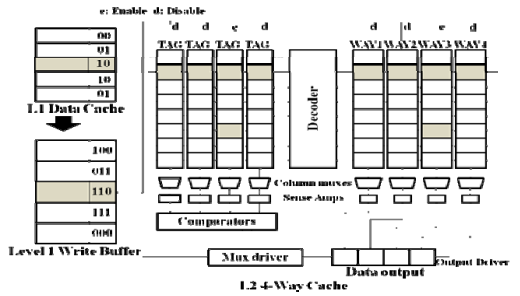
위의 단점들을 극복하기 위해 본 연구 에서는 내장형 시스템을 위한 2-레벨 캐쉬 구조를 제안한다. 제안된 2-레벨 캐쉬를 그림1의 (b)와 (c)에 나타내었다. L1 캐쉬는 직접사상, 바로쓰기 1KB의 크기의 특징을 갖는다. <표2>와 같이, 1KB의 직접사상 캐쉬는 집합연관 캐쉬에 비해 절반 가량의 접근 시간을 보여준다. 따라서, L1 캐쉬는 한 사이클의 접근 시간을 가정할 수 있다. 또한 작고 직접사상을 채용한 캐쉬는 <표2>와 같이 에너지 소비 측면에서도 효율적이다. 그러나 그 작은 사이즈와 낮은 연관정도로 인해 L1 캐쉬는 미스율을 증가시킨다. 이 문제를 극복하기 위해서 제안된 L2 캐쉬의 용량은 일반적인 1-레벨 캐쉬 구조의 크기처럼 유지된다. 본 논문에서는 2가지 형태의 2-레벨 캐쉬 구조를 제안한다. 그림 1 (b)에서 명령어 캐쉬는 1-레벨 구조의 일반적인 크기인 반면 데이터 캐쉬의 L1 캐쉬는 작은 크기이며 L2 데이터 캐쉬는 일반적인 캐쉬 크기를 갖는다. 그림 1 (c)에서는 L1 명령어와 데이터 캐쉬가 작은 크기와 직접사상이며 L2는 통합된 형태의 집합연관 캐쉬이다. L2 캐쉬는 일반적인 캐쉬 구조의 1-레벨 명령어와 데이터 캐쉬를 통합한 크기를 갖는다.

L1 캐쉬와 L2 캐쉬 사이에는 쓰기버퍼가 있으며 메모리 접근을 최소화 하기 위해 L2 캐쉬는 나중쓰기를 채용하였다. L1 캐쉬와 L2 캐쉬 사이에는 no-write allocation정책을 채용했으며 L2 캐쉬와 메모리 사이에는 write allocation정책을 채용하였다.

직접사상과 바로쓰기 그리고 no-write allocation 정책을 사용할 때 L1 캐쉬에서 L2 캐쉬로 쓰기 연산의 전력소모를 줄이는 것이 가능하다. 그림 2와 같이, 쓰기연산에서 L2 캐쉬의 웨이 번호를 저장하기 위해 L1 캐쉬의 각 셋에 2비트와 쓰기 버퍼 각 엔트리에 3비트가 추가되었다. 데이터가 L2에서 L1으로 로드될 때 L2 캐쉬에 저장된 데이터의 웨이 번호가 L1 캐쉬의 추가된 비트에 할당된다. 그러면 L1 캐쉬에서 쓰기 히트일 경우 inclusive 특성을 가정하면, L2 캐쉬에 접근할 때 히트된 데이터는

이전에 저장된 추가 비트에서 지정하는 웨이에 위치하게 된다. 이때 태그비트를 비교하는 것은 필요하지 않다. 이 기법을 본 연구에서는 One-way 접근이라 부르기로 한다. 반대로, 쓰기연산일 경우 L1에서 미스가 발생하면 no-write allocation정책을 가정했기 때문에 데이터 쓰기는 L1 캐쉬가 아니고 쓰기버퍼에만 행해진다. 추가된 비트수는 아래의 공식으로 계산할 수 있다. 1KB L1 캐쉬는 32셋이며 쓰기 버퍼는 8개의 엔트리를 가지고 있다. 쓰기 버퍼의 추가된 1비트는 One way로 쓰기를 할 것인지 아닌지 결정하기 위해 추가되었다.

Log_2 (Number of L2 cache ways) # of L1 data cache sets + (Log_2 (Number of L2 cache ways)+1) * # of write-buffer entries*



(그림 2) L2 캐쉬를 위해 사용된 One-way 접근 기법

4. 결 과

4.1 실험 환경

제안된 2-레벨 캐쉬 구조를 평가하기 위해서 Xscale을 모델링한 XTREM 시뮬레이터[8]가 사용되었다. XTREM은 사이클 단위로 성능을 측정하기 위한 Sim-XScale과 Wattch[10]를 기반으로 한 파워 모델로 구성되어 있다. XTREM에서 사용한 캐쉬 파워 모델은 높은 연관 캐쉬를 위한 CAM-based 모델로서 제안된 캐쉬구조와 맞지 않기 때문에 파워 소비를 계산하기 위해 <표2>의 0.18um 공정의 CACTI 3.2[7]로 모의 실험 하였다. 제안된 2-레벨 캐쉬 구조의 이득을 평가하기 위해 Mibench[5]와 MPEG2 디코더로 12개의 벤치마크를 선택하였다.

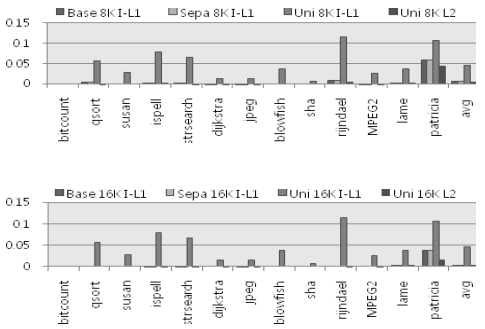
<표 3> 평가된 캐쉬 구조들

Name	Configuration (L1-I$/L1-D$/L2$)
Base8K	8KB 4-way / 8KB 4-way WB
Base16K	16KB 4-way / 16KB 4-way WB
Sepa8K	8KB 4-way / 1KB 1-way WT/ 8KB 4-way WB
Sepa16K	16KB 4-way/1KB 1-way WT/16KB 4-Way WB
Uni8K	1KB 1-way/1KB 1-way WT/16KB 4-way WB
Uni16K	1KB 1-way/1KB 1-way WT/32KB 4-way WB

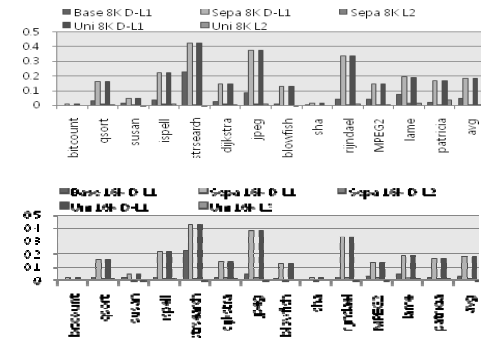
<표3>는 각 캐쉬 구조들의 명명된 이름과 구성을 보여준다. 그림 1의 (a), (b), (c)는 각각 Base, Sepa, Uni로 명명 되었다. WT는 바로쓰기(write-trough)를 나타내며 WB은 나중쓰기 (write-back)를 나타낸다. 1KB WT L1 캐쉬의 읽기/쓰기 연산은 1사이클로 나머지 집합연관 캐쉬의 읽기 연산을 위해 2사이클 그리고 쓰기 연산을 위해 3사이클을 가정하였다.

4.2 실험 결과

그림 3과 4는 실험된 벤치마크들의 명령어와 데이터 캐쉬미스율을 보여준다. 그림 3에서, 작은 L1 캐쉬의 사이즈(1KB) 때문에 일반적으로 Uni8K/Uni16K는 미스율이 증가하였다. 그러나 L2 캐쉬의 글로벌 미스율은 다른 구조들처럼 낮은 수치를 보여준다. 이것은 Uni8K/Uni16K의 L2 캐쉬가 Base 구조만큼 메모리로의 접근을 차단한 것을 나타낸다. 특히 patricia 벤치마크의 경우, L2 캐쉬의 미스율이 다른 구조들보다 낮은 것은 큰 사이즈(16KB 또는 32KB)의 L2 캐쉬의 이득이 큰 것을 나타낸다. 그림 4에 데이터 미스율을 나타내었다. Sepa8K/Uni8K와 Sepa16K/Uni16K의 경우 같은 L1 캐쉬 크기(1KB)로 인해 미스율이 같음을 확인 할 수 있다. 데이터 캐쉬 미스율에서 많은 경우에 Sepa와 Uni 구조가 낮은 글로벌 미스율을 나타낸다. 따라서 2-레벨 캐쉬가 데이터 캐쉬에도 효과적이라 할 수 있겠다.

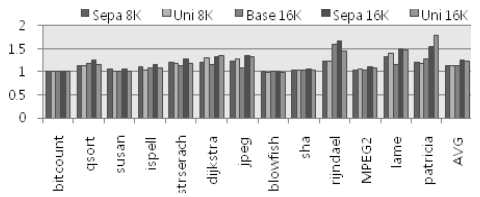


(그림 3) 8K 구조의 명령어 미스율 (위)과 16K 구조의 명령어 미스율(아래)

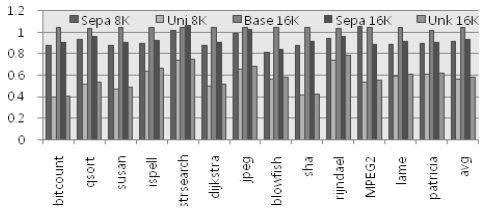


(그림 4) 8K 구조의 데이터 미스율 (위)과 16K 구조의 데이터 미스율(아래)

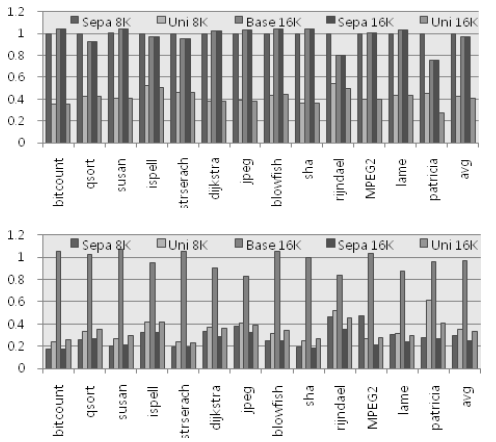
사이클 기반의 성능 결과가 그림 5에 있다. Sepa/Uni의 성능결과는 Base8K로 정규화 되었다. 전체적인 성능이 제안된 2-레벨 캐쉬에서 향상되었음을 확인할 수 있다. 평균적인 성능은 Sepa8K/Uni8Kd 구조에서 14%/14% 그리고 Sepa16K /Uni16K에서 10%/8% 향상되었다. 그림 6은 명령어와 데이터 캐쉬에서 소비된 파워를 나타낸다. 소비된 파워를 계산하기 위해서 <표2>에 있는 값과 시뮬레이션을 통해 세어진 각각의 캐쉬에 접근한 횟수와 히트한 횟수, 미스한 횟수를 이용하였다. 그림 6에서와 같이 Sepa/Uni 구조의 경우 작은 크기의 L1 캐쉬와 One-way 기법으로 인해 전체적인 에너지 소비가 감소하였다. 특히 Uni8K/Uni16K의 경우, 에너지 소비가 급격하게 감소하는 것을 확인할 수 있다. 평균적인 에너지 소비는 Uni/Sepa에서 43%/8% 감소 하였다.



(그림 5) Base8K로 정규화된 사이클 기반의 성능측정결과



(그림 6) Base8K로 정규화된 캐쉬 에너지 소비



(그림 7) Base8K로 정규화된 Energy-Delay Product 명령어 캐쉬(위) 와 데이터 캐쉬(아래)

명령어와 데이터 캐쉬의 Base8K로 정규화된 에너지-지연 곱의 결과(Energy Delay Product: EDP)가 그림 7에 나타나 있다. EDP를 계산하기 위해 캐쉬 에너지 소비와 AMAT(Average Memory Access Time)값이 사용되었다. 평균적인 EDP 감소는 명령어의 경우 Uni/Sepa 구조가 58%/2%이며 데이터를 위해 66%/73%이다. 명령어 캐쉬의 경우 Uni 구조가 가장 향상된 EDP값을 보여주었다. 데이터 캐쉬의 경우 통합된 L2 캐쉬에서 명령어와 데이터간 충돌 발생으로 인해 Sepa 구조가 Uni 구조보다 나은 결과를 보여주었다.

5. 결 론

본 연구에서는 에너지-지연 곱의 측면에서 효율적인 2-레벨 캐쉬 구조를 제안하였다. 제안된 2-레벨 캐쉬구조는 캐쉬의 용량을 효율적으로 사용함으로써 낮은 접근시간과 낮은 글로벌 미스율을 보여주었다. 제안된 캐쉬 구조를 평가하기 위해 사이클 정확도의 시뮬레이터와 다양한 벤치마크를 사용한 실험을 수행하였다. 제안된 2-레벨 캐쉬 구조는 빠른 L1 캐쉬 접근과 L2 캐쉬를 이용한 메모리로의 접근 억제를 통해 AMAT을 감소시켰으며 감소된 AMAT은 전체적인 시스템의 성능을 향상시켰다. 추가로 작은 L1 캐쉬는 캐쉬메모리의 에너지 소비를 급격하게 감소시켰다.

6. 참고문헌

[1] Johnson Kin., Munish G., William H., “The Filter Cache: An Energy Efficient Memory Structure”, In Proc. International Symposium on Microarchitecture, 1997.

[2] Weiyu Tang, Rajesh Gupta, Alexandru Nicolau, “Design a Predictive Filter Cache for Energy Savings in High Performance Processor Architecture”, In Proc. International Conference on Computer Design, 2001.

[3] Nikolaos Bellas, Ibrahim Hajj, and Constantine Polychronopouos, “Using dynamic cache management techniques to reduce energy in a high-performance processor”, In Proc. International Symposium on Low Power Electronics and Design, 1999.

[4] Aviral Shrivastava, Ilya Issenin, Nikil Dutt, “A Compiler-in-the-Loop Framework to Explore Horizontally Partitioned Cache Architectures”, In Proc. Asia and South Pacific Design Automation Conference, 2008.

[5] Matthew R. Guthaus, Jeffrey S. Ringenberg, Dan Ernst, Todd M. Austin, Trevor Mudge, Richard B. Brown, “MiBench: A free, commercially representative embedded benchmark suite”, http://www.eecs.umich.edu/mibench/.

[6] Norman P. Jouppi, “Cache Write Policies and Performance”, In Proc. International symposium on Computer Architecture, 1993.

[7] Premkishore Shivakumar and Norman P. jouppi, “CACTI 3.0: An Integrated Cache Timing, Power, and Area Model”, WRL Research Report (Feb. 2001).

[8] Gilberto Contreras, Margaret Martonosi, Jinzhan Peng, Roy Ju, Guei-Yuan Lueh, “XTREM: A Power Simulator for the XScale Core”, In Proc. ACM SIGPLAN/SIGBED conference on compilers, Architectures, and Synthesis, 2004.

[9] Intel XScale Microarchitecture, http://www.intel.com.

[10] D. Brooks, V. Tiwari, M. Martonosi, “Wattch: a framework for architectural for architectural-level power analysis and optimizations”, In Proc. International Symposium on High-Performance Computer Architecture, 2000.