Journal of The Korea Society of Computer and Information
Vol. 21 No. 9, pp. 1-9, September 2016

www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2016.21.9.001

Instruction Flow based Early Way Determination Technique for Low-power L1 Instruction Cache

Gwang Bok Kim*, Jong Myon Kim**, Cheol Hong Kim***

Abstract

Recent embedded processors employ set-associative L1 instruction cache to improve the performance. The energy consumption in the set-associative L1 instruction cache accounts for considerable portion in the embedded processor. When an instruction is required from the processor, all ways in the set-associative instruction cache are accessed in parallel. In this paper, we propose the technique to reduce the energy consumption in the set-associative L1 instruction cache effectively by accessing only one way. Gshare branch predictor is employed to predict the instruction flow and determine the way to fetch the instruction. When the branch prediction is untaken, next instruction in a sequential order can be fetched from the instruction cache by accessing only one way. According to our simulations with SPEC2006 benchmarks, the proposed technique requires negligible hardware overhead and shows 20% energy reduction on average in 4-way L1 instruction cache.

▸ Keyword : Instruction Cache, Low-power, Embedded Processor, Gshare Predictor

I. Introduction

최근 임베디드 프로세서에서는 성능향상을 위해 1차 명령어 캐쉬에 주로 높은 연관사상 구조를 사용하고 있다. 연관사상 캐쉬 구조는 직접사상 방식에 비해 더 많은 전력을 소비하는데, 임베디드 프로세서가 주로 적용되는 모바일 디바이스는 배터리를 사용하기 때문에 효율적인 전력 사용이 매우 중요하다. 최근 인텔의 자료에 따르면 프로세서 코어의 전체 전력 중 12~45%가 캐쉬에서 소모된다[1]. 특히, 명령어인출 단계에서 소모되는 전력은 ARM Cortex-A15의 경우 전체 프로세서 코어의 전력소모 중 12-15%가량을 차지하고 있다[2]. 명령어 인출을 담당하는 1차 명령어 캐쉬의 경우 매 사이클마다 접근이 이루어지므로 저전력 기법적용이 매우 중요하다. 그림 1은 복잡한 연산 위주의 컴퓨팅과 그렇지 않은 경우 각각에 대해 코어의 각 부분별 전력소모 비율을 보여준다. 복잡한 연산이 수행되지 않는 경우 캐쉬가 차지하는 전력소모는 45%에 달하는 것을 확인할 수 있다.

이러한 이유로, 연관사상을 사용한 캐쉬에서의 불필요한 전력소모를 줄이기 위한 많은 연구가 수행되고 있다. 캐쉬 구조는 크게 태그 배열과 데이터 배열 부분으로 구분할 수 있는데 태그 배열이 차지하는 하드웨어 공간과 전력 소모는 데이터 배열이 차지하는 부분보다 상대적으로 매우 적다. 따라서, 데이터 배열에서의 전력 소모를 감소시키기 위한 연구가 주로 이루어지고 있다.

• First Author: Gwang Bok Kim, Corresponding Author: Cheol Hong Kim
*Gwang Bok Kim(loopaz63@gmail.com), School of Electronics and Computer Engineering, Chonnam National University
**Jong Myon Kim(jmkim07@ulsan.ac.kr), School of Electrical Engineering, University of Ulsan
***Cheol Hong Kim(chkim22@chonnam.ac.kr), School of Electronics and Computer Engineering, Chonnam National University
• Received: 2016. 07. 12, Revised: 2016. 08. 11, Accepted: 2016. 08. 28.
• This research was supported by the MSIP(Ministry of Science, ICT and Future Planning), Korea, under the ITRC(Information Technology Research Center) support program (IITP-2016-R2718-16-0011) supervised by the IITP(Institute for information & communications Technology Promotion).

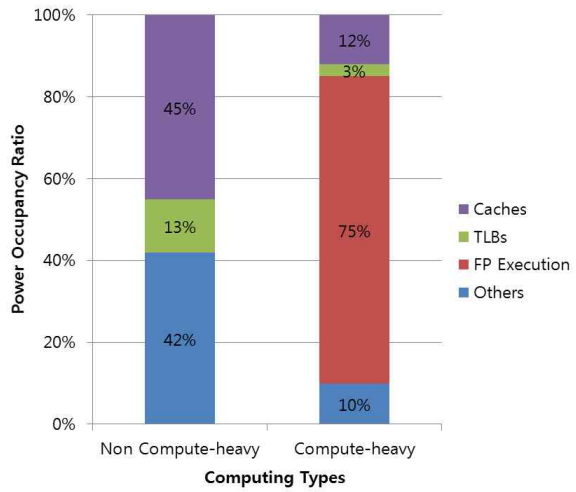

Fig. 1. Power distribution of processor core

일반적인 캐쉬는 전체 접근 사이클 감소를 위해 태그 배열과 데이터 배열을 동시에 접근한다. 연관사상 캐쉬의 경우, 태그 배열과 데이터 배열의 모든 웨이를 동시에 접근하고, 적중되는 경우에는 최종적으로 하나의 웨이에 저장된 데이터를 사용하게 된다. 이 때, 모든 웨이를 활성화시킴으로 인해 불가피하게 큰 전력소모가 발생한다.

그림 2는 태그배열, 데이터배열, 디코더, 비교기 등으로 간략화된 일반적인 캐쉬구조를 보여준다. 디코더를 통해서 선택된 태그배열과 데이터배열을 동시에 접근하고 활성화된 모든 웨이 중 태그가 일치하는 데이터만을 인출한다. 만약 태그 배열을 보다 빨리 접근하여 어떤 웨이를 접근할 지 미리 알 수 있다면 데이터 웨이를 접근하기 위한 비트라인을 모두 활성화하지 않고 하나의 선택된 웨이만을 활성화 시킬 수 있으므로 전력 소모를 크게 줄일 수 있다.

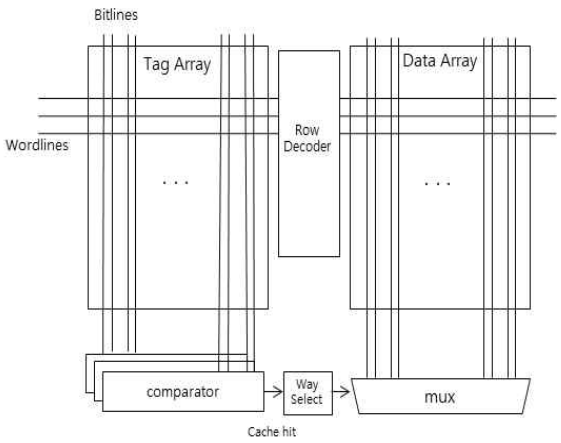

Fig. 2. Traditional cache model

본 논문에서는 1차 명령어 캐쉬에서 소모되는 전력을 감소시킴으로써 전체 프로세서의 전력효율성을 향상시킬 수 있는 기법을 제안하고자 한다. 제안하는 기법은 한 사이클 먼저 예측된 명령어주소에 대한 태그 비교를 수행한다. 한 사이클 먼저 이루어진 태그 비교 정보를 활용하여 실제 데이터 인출은 하나의 웨이에만 접근하고 나머지 웨이는 활성화시키지 않음으로써 전력소모를 감소시킨다. 제안하는 이른 웨이결정기법은 추가적인 하드웨어 오버헤드가 적고, 성능에 거의 영향을 미치지 않는다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구와 이른 분기방향 결정기법을 제안한 연구들을 소개한다. 3장에서는 본 논문에서 제안하는 예측된 분기 방향을 기반으로 하는 이른 웨이 결정 기법을 기술하고, 4장에서는 정성평가와 모의실험결과를 비교분석한다. 마지막으로 5장에서는 본 논문의 결론과 향후 연구계획에 대해서 기술한다.

II. Related Work

1. Low-power Cache Schemes

캐쉬 메모리의 전력소모를 줄이기 위해서 최근까지 많은 기법들이 제안되어 왔다. 특히 높은 연관사상 캐쉬의 웨이를 선택적으로 접근함으로써 불필요한 전력 소모를 감소시키기 위해서 많은 연구들이 수행되었다[3-14].

TLC[3]는 캐쉬와 동시에 접근하는 TLB의 구조를 변경하여 캐쉬에 존재하는 각 라인의 웨이정보를 TLB에 최소화하여 저장하는 방법을 제안하였다. 이 기법에서는 캐쉬의 태그부분을 완전히 제거하고 캐쉬에 존재하는 라인에 대한 존재 유무와 웨이 정보를 TLB에 기록한다. 또한 가상주소를 물리주소로 변환시키는데 필요한 비트수를 줄임으로써 크지 않은 동적 전력으로 TLB에서 웨이를 결정할 수 있다. 하지만 이 기법은 캐쉬와 TLB가 동시에 접근되지 않기 때문에 데이터 웨이에 대한 접근이 VIPT방식보다 접근시간이 길어지게 되고, 물리주소를 활용하지 않음으로써 추가적으로 발생하는 문제점이 있다. TLB와 캐쉬의 상관성이 커짐에 따라 적중실패율이 증가하는 문제를 캐쉬에 존재하는 라인의 수를 고려한 교체정책으로 해결하고, 캐쉬 접근에 필요한 사이클이 늘어나는 문제를 해결하고자 Micro 페이징기법 등을 적용한다.

Early Tag Lookup[5]은 다음 PC를 예측하여 접근할 캐쉬의 웨이를 결정하기 위해 PHT와 BTB를 동시에 접근한다. 이러한 동작을 구현하기 위해 캐쉬의 태그부분과 BTB의 뱅크를 2배로 증가시킨다. 하지만 결과적으로 태그 부분의23.91%, BTB의 4.66%, PHT의 4.66% 의 추가적인 하드웨어 공간을 필요로 하게 된다.

1차 데이터 캐쉬의 전력소모를 감소시키기 위한 기법인 ETA(Early Tag Access) 캐쉬[6]는 LSQ 단계에서 태그비교를 수행하여 목표주소에 대한 웨이를 미리 결정한다. 이 기법은 LSQ와 1차 데이터 캐쉬의 동시접근을 사용하는 최신 프로세서와 목적이 같기 때문에 임베디드 시스템에서 더 효율적이다. 이 기법 또한 크지 않은 하드웨어 자원으로 전력 소모를 감소시킬

수 있다. Way-halting[7] 기법은 태그부분을 각각 작은 서브 배열로 나누고, 일부 비트만을 비교하여 캐쉬에 대한 접근을 줄이는 방법으로 불필요한 웨이 접근을 상당히 줄일 수 있다. 하지만 잘못된 예측으로 인한 성능감소와 추가되는 하드웨어 오버헤드를 고려해야 한다.

그 밖에 쓰기연산이 자주 일어나는 L2 캐쉬의 에너지효율을 높이기 위한 기법인 Way-Tagged 캐쉬[8]와 교체대상을 늘려 웨이를 결정하는 캐쉬구조인 Zcache[9] 등이 제안되었다. 또한, Way-Memorization 기법[14]은 CAM-tagged 캐쉬에 대해 웨이 예측이 어려웠던 문제점에 대한 대안을 제시하였다.

2. Early Branch Prediction

분기예측은 다음 명령어주소가 결정되기 이전에 분기 방향 및 다음 명령어주소를 미리 예측하고 실행시켜 전체적인 프로세서의 처리지연시간을 줄이는데 목적을 두고 있다. 만약 현재 명령어주소의 분기방향을 언테이큰으로 예측한다면 BTB 접근이 불필요하지만, 최신 프로세서는 분기방향 예측과 BTB 접근을 동시에 수행하기 때문에 BTB는 항상 접근되어야 한다. 이러한 불필요한 BTB 접근을 줄이기 위해 선택적으로 BTB를 접근하고자 기존의 프로세서의 분기 예측 시간보다 이른 사이클에 분기 예측을 수행할 수 있는 기법이 제안되었다[15]. 선택적인 BTB(Branch Target Buffer) 접근 기법은 분기방향을 예측하는 PHT의 구조를 변경함으로써 현재 명령어주소에 대한 분기방향 예측뿐만 아니라 다음 명령어주소에 대한 분기예측을 위한 룩업(lookup)을 수행한다. 이와 같이 분기예측시간을 앞당겨 BTB를 선택적으로 접근하기 때문에 결과적으로 BTB에서 소모되는 전력을 크게 줄일 수 있다.

그림 3과 같은 구조를 갖는 Gshare 분기방향 예측기는 전역 분기 패턴이 기록되어 있는 글로벌 히스토리와 현재 명령어주소를 XOR 연산한 결과를 인덱스로 하여 PHT의 엔트리에 접근한다. 만약 이전 명령어가 분기하지 않았다면 글로벌 히스토리는 변경되지 않으므로 현재 명령어의 분기예측을 위한 PHT 인덱스 생성은 글로벌 히스토리에 영향을 받지 않게 된다. 오직 PC에 의해 PHT 인덱스가 바뀌게 되고 분기되지 않는 명령어라면 다음 명령어 주소는 PC+8와(64비트 프로세서의 경우) 같은 순차적인 주소라고 예측할 수 있다. 따라서, 분기만 발생하지 않는다면 모든 명령어가 1사이클 빨리 PHT 접근을 할 수 있기 때문에 이른 분기방향 예측이 가능하게 된다. 물론 BTB 적중이나 분기가 발생한 경우는 히스토리의 값이 바뀌므로 변경된 히스토리와 PC를 사용하여 다시 PHT에 접근해야 한다. 만약 다음 명령어의 주소가 현재 주소에 대하여 순차적 주소로 예상된다면 그림 3과 같이 마지막 비트만 다른 두 개의 엔트리를 묶은 PHT 구조를 만들고 이를 접근할 때마다 항상 두 개의 결과를 인출할 수 있다. 이는 기본 PHT 구조와 비교하여 한 번 접근할 때마다 소모되는 전력은 약 4%정도 더 필요하지만 PHT에 접근하는 수는 절반가량으로 감소되기 때문에 전체 전력은 오히려 줄어든다[25]. 또한, 아주 작은 하드웨어 오버헤드만을 필요로 하며 성능감소는 거의 없다.

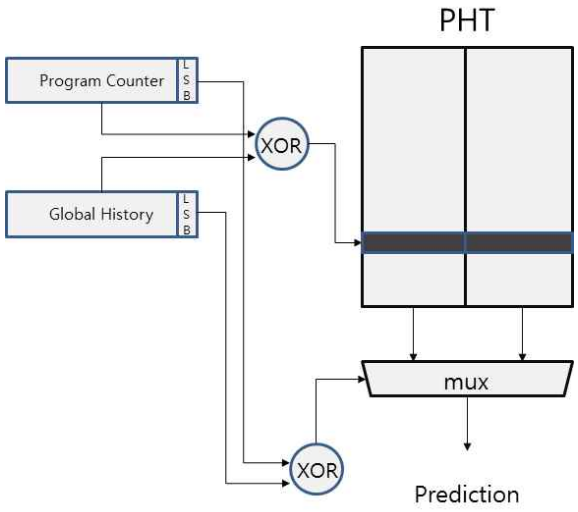

Fig. 3. Gshare predictor for early branch prediction

III. The Early Way Determination Technique

본 장에서는 이른 PHT 접근을 기반으로 한 분기예측에 따라 다음 명령어주소에 해당되는 웨이를 결정하는 기법을 기술한다. 일반적인 연관사상 캐쉬에서는 태그와 데이터배열이 동시에 접근된다. 캐쉬의 모든 데이터 웨이를 활성화하여 접근할 때에는 어떤 웨이가 해당되는지 태그 비교가 수행되지 않은 상태이고 요청된 블록이 캐쉬에 존재하는지도 알 수 없다. 본 논문에서는 1차 명령어 캐쉬에 대하여 모든 웨이를 접근하기 전에 요청한 블록이 존재하는 웨이를 결정하는 이른 웨이 결정 기법을 제안한다.

1. Sequential Access Mode

본 논문에서 제안하는 기법은 순차접근모드를 우선적으로 확인하고 동작한다. 이전 명령어가 분기를 발생시키는 경우를 제외한 모든 명령어는 이른 사이클에 PHT 접근이 가능한데, 이러한 경우를 만족시키는 상태를 본 논문에서는 순차접근모드라고 명명한다. 또한, 순차적인 명령어주소는 명령어가 64비트의 길이를 가졌을 때로 가정하여 PC의 다음 주소를 PC+8로 한다.

그림 4는 제안하는 기법에 대한 순차접근모드의 동작과정을 보여준다. PHT에 접근하기 위한 인덱스는 Gshare 분기예측기에 따라 명령어주소인 PC와 글로벌 히스토리에 대해 XOR 연산을 통해 구할 수 있다. 본 논문에서는 앞서 기술했던 PHT 구조[25]를 구현하여 동시에 두 개의 분기예측 결과를 가져온다. 두 개의 분기 예측결과는 오직 마지막 비트만 서로 다른 두 개

의 인덱스로 접근하므로 PHT 접근을 위한 인덱싱에서는 마지막 비트를 제외한다. 이 후, 인출단계에서 현재 PC의 마지막 비트와 글로벌 히스토리의 마지막 비트를 XOR 연산하여 명령어에 대한 분기방향 예측을 수행한다. 또한, 분기방향 예측과 함께 순차접근모드가 활성화되며 BTB 적중과 같은 분기가 발생한다면 순차접근모드를 종료한다. 모든 명령어는 이른 분기방향 예측과 함께 순차접근모드일 경우에만 다음 명령어에 대한 웨이 결정이 가능하다. 예를 들어, n번째 명령어의 주소에 대한 캐쉬웨이를 1사이클 빨리 결정하고자 한다면 n-1번째 명령어가 순차접근모드인 상태이고, 이를 위해서는 n-2번째 명령어 또한 분기되지 않아야 한다. n-1번째 명령어가 인출되는 순간 순차접근모드가 활성화된 상태라면 다음 명령어주소로 예측되는 PC+8로 캐쉬의 태그배열에 접근하여 웨이를 결정하고 다음 명령어의 파이프라인을 위해 웨이 위치정보를 저장한다.

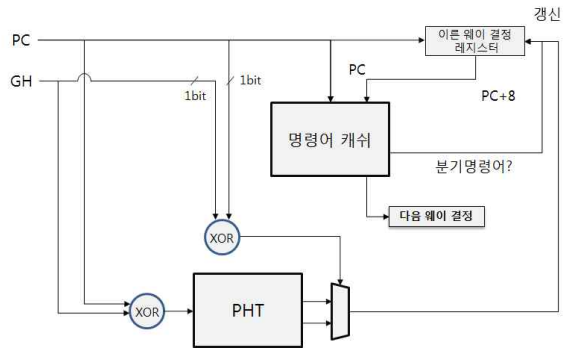

Fig. 4. Procedure for sequential access mode

2. Early Way Determination

본 논문에서는 다음에 수행될 명령어주소에 대한 웨이를 1사이클 빠르게 결정하기 위해서 순차접근모드의 동작을 우선적으로 확인한다. 만약 현재 상태가 순차접근모드라면 현재 명령어에 대한 분기예측은 이미 1사이클 먼저 수행되었다는 것을 의미하고, 해당 분기예측 결과가 언테이큰이라면 다음 명령어 주소인 PC+8에 대한 웨이 결정을 수행한다.

그림 5는 순차접근모드 제어와 웨이결정에 대한 정보를 공유하기 위해 추가하는 이른 웨이결정 레지스터 구조를 나타낸다. 이른 웨이 결정 레지스터는 순차접근모드비트 1비트와 이른 접근을 확인하기 위한 EA(early access) 비트, 요청블록의 존재여부를 나타내는 1비트의 valid 비트, 마지막으로 결정된 웨이위치를 저장하는 NPC_way(Next PC way) 비트로 구성된다. NPC_way 비트는 명령어 캐쉬의 웨이수에 따라 달라지는데, 본 논문에서는 4웨이의 연관사상 방식의 1차 명령어 캐쉬를 대상으로 하기 때문에 NPC_way는 3비트로 구성한다. 그러므로, 제안된 기법을 구현하기 위해 추가되는 하드웨어는 전체 프로세서의 공간에 비해 아주 작은 수준이라고 할 수 있다.

순차접근모드 비트는 BTB가 적중되거나 명령어 해독 결과가 분기명령어로 판명된 경우에 0으로 저장된다. BTB 적중은 분기를 발생시키고 분기 위치는 PC+8 가 아닌 BTB에서 예측한 주소로 결정한다. 따라서 이 경우에는 다음 명령어주소를 1사이클 빨리 알 수 없기 때문에 순차접근모드 비트를 0으로 함으로써, 다음 명령어에 대한 이른 웨이 결정을 하지 않도록 한다. 명령어 해독 후 분기명령어로 판명된 경우 또한 순차접근모드 비트는 0으로 저장된다. 만약 현재 명령어가 분기명령어라면 무조건 글로벌 히스토리를 갱신하기 때문에 다음 명령어가 PHT에 접근하기 위한 인덱스 생성에 영향을 미친다. 결과적으로, 이 두 경우에 대해서 순차접근모드를 종료시킴으로써 이어지는 다음 웨이예측을 제한할 수 있다.

EA 비트는 현재 명령어주소를 이용하여 1사이클 전에 캐쉬에 접근하여 태그 비교를 이미 수행하였는지를 나타낸다. 다시 말해, 현재 명령어주소에 대해 이미 태그비교를 수행하였는지를 확인하는 비트이며, 확인된 비트에 따라 현재 명령어주소에 대해 태그비교를 수행하거나 수행하지 않는다. 순차접근모드상태에서 다음 명령어에 대해 1사이클 빨리 접근한다면 EA 비트를 1으로 저장하고 분기가 발생하거나 분기예측기가 테이큰의 결과를 보인다면 EA 비트는 0으로 저장된다. 만약 EA 비트가 1로 되어있다면 현재 명령어주소는 이미 이른 사이클에서 태그비교를 거쳤음을 나타내고 다른 비트를 즉시 확인한다. 만약 EA 비트가 0으로 되어있다면 결정된 웨이가 없을 뿐만 아니라 이른 명령어 캐쉬 접근조차 없었음을 나타내기 때문에 기존 프로세서와 동일하게 캐쉬에 접근한다.

Valid 비트는 이른 캐쉬접근이 발생한 경우 요청된 블록이 캐쉬에 존재하는지 여부를 나타낸다. 만약 한 명령어의 이른 웨이 결정을 위해 1사이클 빨리 접근했음에도 불구하고 캐쉬에 요청블록이 존재하지 않아 적중실패가 발생한다면 캐쉬의 데이터배열을 활성화시키지 않아도 된다. 다시 말해 valid 비트가 0으로 되어있다면 요청된 블록은 캐쉬에 존재하지 않기 때문에 기존의 프로세서와 마찬가지로 하위 레벨의 캐쉬에 접근한다. 반대로 valid 비트가 1로 되어있다면 NPC_way 비트를 참조하여 결정된 웨이에 접근한다. NPC_way는 웨이의 위치를 나타내는 3비트로 구성되어있으며, 이전 비트가 반드시 먼저 참조되기 때문에 초기화 할 필요가 없다.

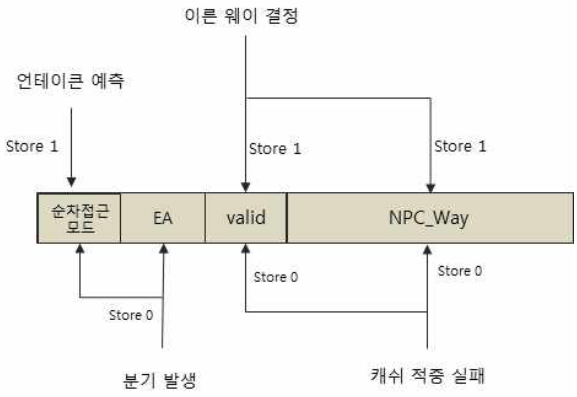

Fig. 5. Register for early way determination

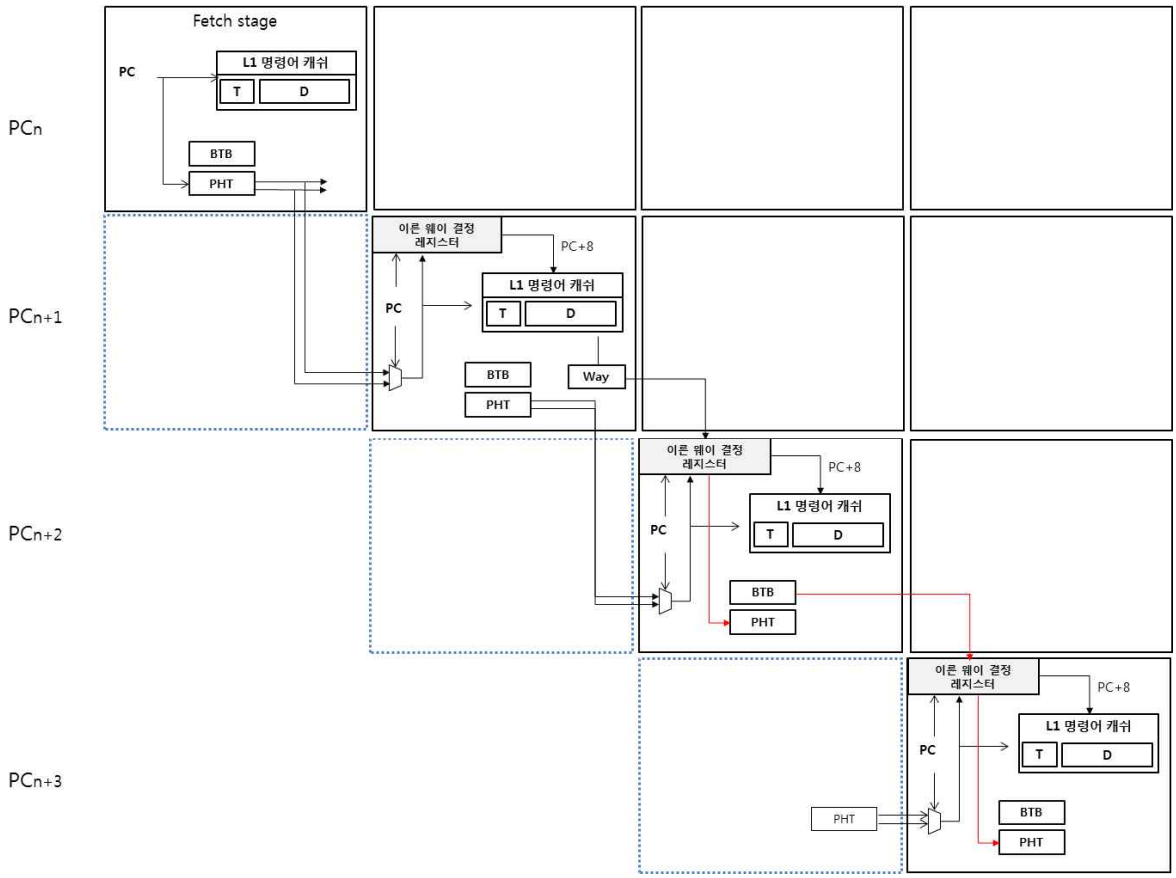

Fig. 6. Proposed early way determination

그림 6은 이른 웨이 결정 기법이 적용된 프로세서의 명령어 수행 흐름을 보여준다. 첫 번째 명령어주소에 대해서는 기존의 프로세서와 동일하게 명령어 캐쉬에 접근한다. PHT로부터 현재 명령어 주소인 PC와 다음 순차 명령어주소인 $PC+8$에 해당하는 두 개의 분기예측을 무조건적으로 가져온다. 만약 BTB 적중이나 명령어 해독결과가 분기명령어가 아니라면 순차접근모드는 유지되기 때문에 다음 명령어는 성공적으로 이른 분기 방향예측이 가능하다. 두 번째 명령어의 주소 PC_{n+1}은 가장 먼저 이른 웨이 결정 레지스터를 확인한다. 만약 순차접근모드에 진입된 상태라면 즉시 EA 비트를 확인하여 PC_{n+1}의 웨이가 결정되었는지 확인한다. EA 비트가 1로 되어있다면, 태그를 비교하지 않고 결정된 웨이에 해당하는 데이터만을 인출한다. 이와 동시에 분기방향 예측결과를 MUX를 통해 즉시 알 수 있는데, PC_n에 해당하는 명령어는 어떠한 분기도 발생시키지 않았으므로 순차접근모드가 활성화된 상태로 유지된다. 따라서, PC_{n+1}의 이른 분기방향 예측결과가 언테이큰이라면 가장 먼저 레지스터의 순차접근모드상태를 확인해야 한다.

순차접근모드가 활성화되었음을 확인하면, 다음 명령어주소로 예측되는 $PC_{n+1}+8$을 이용하여 명령어 캐쉬에 한 번 더 접근한다. 명령어 캐쉬에 재접근하는 경우 데이터를 요청하지 않고 오직 태그비교기를 통해 요청블록과 일치하는 웨이를 결정한다.

그림 6의 마지막 PC_{n+3}는 이전 명령어가 분기를 발생시킨 경우를 보여준다. 만약 PC_{n+2}와 같이 BTB가 적중된 경우 순차접근모드를 빠져나오기 때문에 다시 순차접근모드가 되기 전까지 이른 웨이 결정을 수행하지 않는다. 이전 명령어인 PC_{n+2}는 BTB가 적중되거나 분기명령어로 판명된 경우 글로벌 히스토리를 갱신하기 때문에 PHT에 재접근 해야한다. 따라서 순차접근모드를 중지하고 이른 웨이 결정 또한 수행하지 않는다. PC_{n+2}가 분기를 발생시키기 전에 수행된 PC_{n+3}의 이른 웨이 결정은 잘못된 결과이므로 EA 비트와 valid 비트 모두 0으로 변경한다. 이와 동시에 변경된 글로벌 히스토리와 PC를 이용하여 PHT에 대한 인덱스를 다시 계산하여 접근한다. 이 때 PHT에서 예측결과는 또 다시 두 개의 결과를 인출하기 때문에 그림에 표기되지 않은 PC_{n+3}의 다음 명령어 주소에 대한 예측 또한 이른 사이클에서 수행하게 된다.

IV. Experiments

이 장에서는 제안된 이른 웨이 결정 기법에 따른 소모 전력을 정성적으로 분석하고 Gshare를 사용하는 기존 구조에서의 소모전력과 비교한다. 실험을 위해 SimpleScalar[16] 시뮬레

이터와 Wattch[17]를 사용한다. SimpleScalar는 실제 프로세서가 구동되는 것과 같은 결과를 얻을 수 있는 시뮬레이터이며 Wattch는 모델링된 프로세서 구조에 따라 상세한 전력소모를 계산한다. 실험의 입력으로는 SPEC2006 Benchmarks[18]를 사용한다. 모의실험 시 사용한 프로세서의 구성 변수들은 표 1과 같다.

Table 1. Parameters of simulated processor

Parameter	Value
Branch predictor	Gshare predictor PHT/4KB
BTB	2K-entry, 2way
L1 I-cache	16KB size, 32B line, 4way
L1 D-cache	32KB size, 32B line, 4way
L2 Unified cache	1MB size, 64B line, 8way

Table 2. Notation for the analytical model

Notation	Definition
N_{SPP}	순차접근모드 진입상태에서 명령어 캐쉬에 접근하는 명령어 수
N_{TOTAL_INST}	명령어 캐쉬에 접근하는 총 명령어 수
N_{BRANCH_EXE}	분기명령어 수
$P_{I\_CACHE\_DATA}$	명령어 캐쉬의 데이터부분에 대한 총 전력 소모량
P_{TRAD_DATA}	명령어 캐쉬의 데이터배열에 소모전력 (한 번 접근시)
P_{EARLY_DATA}	명령어 캐쉬에서 하나의 웨이만 활성화 된 데이터배열의 소모전력 (한 번 접근시)
N_{TRAD_ACCESS}	모든 웨이가 활성화된 데이터배열에 접근하는 명령어 수
N_{EARLY_ACCESS}	제안기법에 따라 결정된 웨이에만 접근하는 명령어의 수

1. Analysis Methodology

모의실험에 앞서 분석모델을 통해 전력소모량을 예측해 보고자 한다. 본 논문에서 제안하는 이른 웨이결정 기법은 앞서 기술한대로 순차접근모드에서만 수행된다. 전체 명령어 중 순차접근모드가 1인 상태를 가지는 명령어의 수 N_{SPP}는 BTB의 적중정확도를 100%라고 가정하였을 때, 명령어 캐쉬의 접근수 N_{TOTAL_INST}에서 분기명령어의 수 N_{BRANCH_EXE}를 제외한 명령어수와 같다. 따라서 수식으로 표현하면 식 (1)과 같이 표현할 수 있다. 이른 PHT 접근을 통해 두 개의 분기방향을 예측한다면 PHT 접근 수는 기존구조에 비해 절반에 가깝지만[15] 본 논문에서는 캐쉬의 소모전력을 대상으로 하기 때문에 PHT 접근 수와 관계없이 빠른 분기예측이 가능한 명령어만을 고려한다. 다시 말해 PHT를 통해 두 개의 분기예측결과를 이미 인출하고 MUX를 통해 빠른 분기방향예측이 가능한 명령어만을 고려한다.

$$N_{SPP} = N_{TOTAL\_INST} - N_{BRANCH\_EXE} \quad (1)$$

본 논문에서는 모의실험에서의 전력측정을 위해 CACTI[20]의 캐쉬모델을 활용한다. 전력측정을 위해 명령어 캐쉬의 데이터부분에 대한 전력계산식을 간략화하면 다음 식 (2)와 같다.

$$P_{I\_CACHE\_DATA} = P_{TRAD\_DATA} \times N_{TRAD\_ACCESS} + P_{EARLY\_DATA} \times N_{EARLY\_ACCESS} \quad (2)$$

명령어 캐쉬의 데이터부분에 대한 전체 전력 $P_{I_CACHE_DATA}$은 두 경우에 대한 전력의 합으로 이루어진다. 제안된 기법이 적용되지 않은 기존의 프로세서는 캐쉬의 모든 웨이에 접근한다. 따라서 기법이 적용되지 않은 명령어에 대한 캐쉬전력은 한 번 접근시 소모되는 P_{TRAD_DATA}와 해당 접근수의 곱으로 표현할 수 있다. 두 번째로, 하나의 웨이만 활성화된 캐쉬의 전력소모는 한 번 접근 시 소모되는 전력 P_{EARLY_DATA}와 해당 접근 수의 곱으로 계산할 수 있다.

Table 3. Branch instruction ratio for the benchmarks

벤치마크	분기명령어 비율
bwaves(CFP)	9%
bzip2(CINT)	3%
gobmk(CINT)	17%
gromacs(CFP)	14%
hmmer(CINT)	8%
lbm(CFP)	12%
mcf(CINT)	4%
milc(CFP)	6%
sjeng(CINT)	4%
zeusmp(CFP)	4%
Average	7.5%

각 벤치마크별 분기명령어의 비율은 모의실험을 통해 직접 측정하였으며 그 결과는 표 3와 같다. 선택된 10개의 벤치마크 중 gobmk, gromacs, lbm이 각각 10% 이상의 분기명령어비율을 보여주고 평균 7.5%의 비율을 보인다. 또한 분기명령어의

개수와 별도로 Gshare 분기예측기의 분기예측 정확도에 따라 분기가 추가적으로 발생할 수 있는데, 선행 연구[19]의 분석결과와 같이 4K 엔트리를 가지는 Gshare 분기예측기의 경우 gobmk 벤치마크에 대해 10% 이상의 높은 예측실패율을 보이지만 gromacs나 zeusmp와 같은 벤치마크의 경우 약 1% 정도의 실패율을 보인다.

2. Experimental Results

본 논문에서는 명령어 캐쉬에서의 전력소모를 측정, 분석하기 위해 SPEC2006 벤치마크를 입력으로 하여 그 전력소모량을 비교한다. 선택된 17개의 벤치마크를 4-웨이를 갖는 명령어 캐쉬와 8-웨이를 갖는 명령어 캐쉬에 대하여 각각 측정한다. 또한 모의실험에서는 명령어 캐쉬의 크기를 16KB으로 고정한다.

그림 7은 전체 명령어 캐쉬의 접근 중 제안하는 이른 웨이결정기법의 성공비율을 나타낸다. 또한 그래프는 각 벤치마크별 다른 캐쉬 웨이수에 따른 모의실험 결과를 보여준다. 웨이별 기법 적용이 성공한 경우의 수는 웨이별로 유사한 결과를 보이는 반면 gobmk와 gromacs 벤치마크의 경우 다른 웨이를 가진 캐쉬에 비해 낮은 성공률을 보인다. 동일한 분기명령어의 수에 따른 순차접근모드 비율이 같고, 캐쉬접근 수 또한 동일하지만 캐쉬의 웨이수가 줄어듬에 따라 캐쉬적중실패가 발생하고 결과적으로 제안하는 기법으로 인한 웨이결정수가 줄어든다.

그림 8에서 보이는 바와 같이 2웨이에서의 전력감소율은 평균 10.8%에 달하고, 본 모의실험에서 기준이 되는 4웨이의 캐쉬에서는 평균 20%의 전력감소를 보인다. 8웨이에서는 더 높은 전력감소율을 보여주지만 감소율간의 증감률은 2웨이와 4웨이간에 보여주었던 증감률보다는 적음을 알 수 있다. 4웨이 캐쉬의 전력감소율은 2웨이 캐쉬에 비해 85% 더 증가한 반면

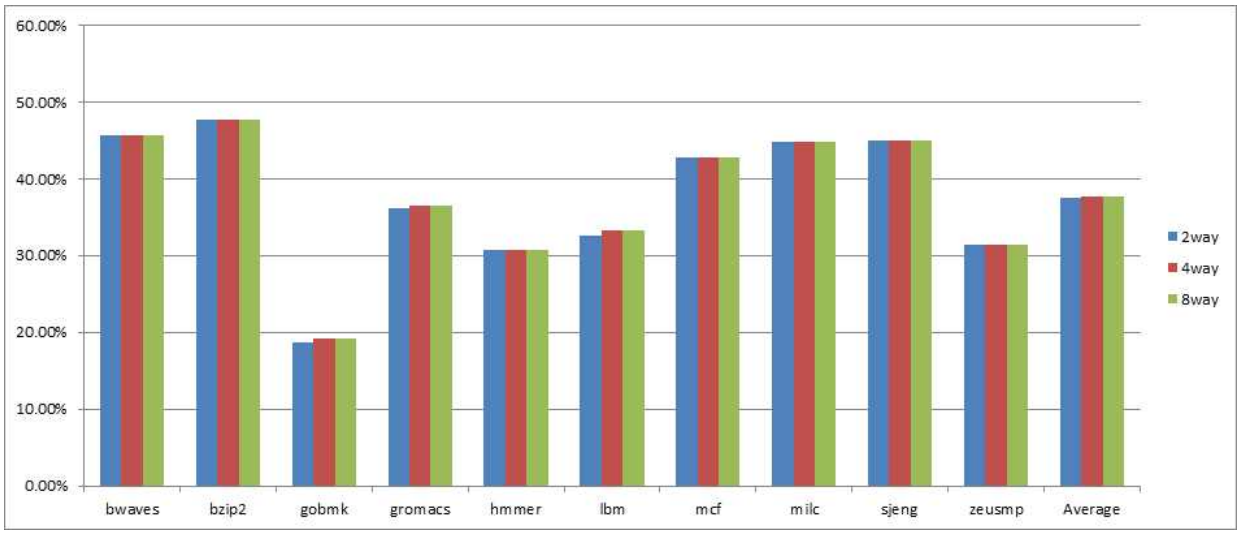

Fig. 7. Number of Instruction for early way determination

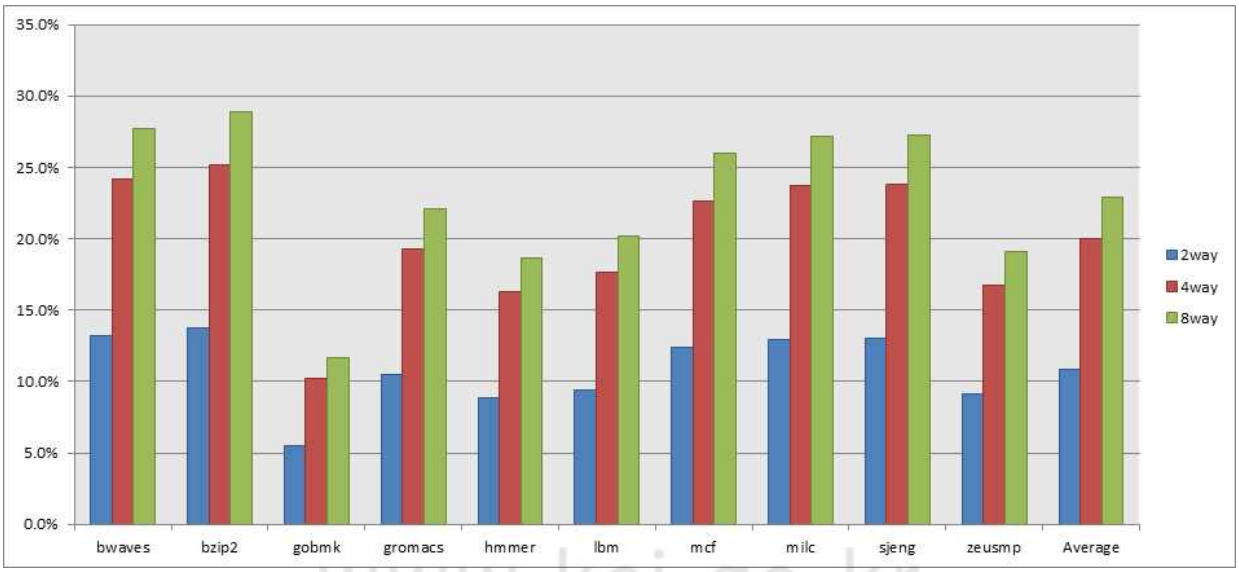

Fig. 8. Power reduction ratio

8웨이는 4웨이 캐쉬에 비해 14% 전력감소율이 증가한다. 거의 동일한 웨이결정 수를 갖게 되지만 한 번에 제거되는 웨이의 수가 다르기 때문에 웨이의 증가에 따라 전력효율 또한 증가하는 것을 확인할 수 있다. 반대로 웨이의 증가로 인한 태그배열의 전력소모는 증가하게 된다. 따라서 웨이 증가에 따른 전력감소율의 증가폭은 점점 감소한다.

그림 8의 벤치마크별 모의실험 결과를 살펴보면, gobmk 벤치마크의 경우 다른 벤치마크보다 낮은 전력감소율을 보인다. 이와 같은 결과는 높은 분기명령어의 비율과 이에 따른 순차접근모드에 해당하는 명령어의 수가 줄어들기 때문으로 분석된다. 분기명령어의 비율은 기존 연구에서도[30] 15% 정도의 비율로 비교적 높은 분기명령어를 포함하였고, 본 논문에서의 측정에서도 약 17%의 분기명령어비율을 보인다. 또한 gobmk 벤치마크의 경우 gshare 분기예측기에 대하여 상대적으로 낮은 정확도를 보여 기법의 효율성이 감소한다. 제안하는 기법의 적용으로 인해 가장 좋은 전력감소를 보여주는 벤치마크는 bzip2으로, 각 웨이별 모의실험에서도 가장 좋은 결과를 보여준다. 그 이유로는, 제안하는 기법의 효율성에 영향을 주는 요소인 분기명령어 비율과 언테이큰 비율로 분석된다. 먼저 분기명령어의 비율이 3%로 가장 낮은 bzip2은 모의실험에 사용한 벤치마크 중 가장 높은 전력 감소 효과를 보여준다. 이와 비슷한 분기명령어 비율을 가진 mcf, sjeng, zeusmp 모두 4%의 분기명령어로 근소한 차이를 보이고 결과 역시 비슷한 전력감소를 보여준다. 하지만 zeusmp 벤치마크의 경우 전력감소율이 비교적 떨어지는 결과를 보여주는데 이는 Gshare 분기예측기로부터 예측되는 언테이큰 비율의 영향으로 분석된다. zeusmp의 언테이큰 비율은 측정결과 37.7%으로 전체 벤치마크의 평균 언테이큰 비율인 41%보다 낮은 수치를 보여준다. 또한 zeusmp와 거의 동일한 분기명령어비율을 가진 벤치마크 mcf, sjeng은 각각 42.88%, 45.06%의 언테이큰 비율을 나타낸다.

V. Conclusions

본 논문에서는 1차 명령어 캐쉬에서의 불필요한 웨이 접근을 제거하여 캐쉬에서 소모되는 전력을 감소시키기 위해 1사이클 빨리 접근할 데이터 웨이를 결정하는 기법을 제안하였다. 제안하는 기법은 Gshare 분기예측기를 사용하는 임베디드 프로세서를 대상으로 변경된 PHT로부터 두 개의 분기방향을 예측한다. 이를 통해 언테이큰으로 예측되는 명령어는 다음 명령어의 웨이를 미리 결정함으로써 캐쉬의 데이터배열을 모두 활성화하지 않고 결정된 하나의 웨이만 접근하게 된다. 다양한 벤치마크를 대상으로 하는 모의실험을 통해 4웨이 명령어 캐쉬의 경우, 성능 감소 없이 평균 20%의 전력소모를 감소시키는 결과를 확인하였다. 따라서, 제안하는 기법을 적용한다면 비교적 전력소모가 많은 1차 명령어 캐쉬에서의 전력소모를 상당부분 감소시킬 수 있을 것으로 기대된다.

향후에는 분기비율이 높은 벤치마크에서 제안 기법의 효율성을 높임으로써 더 큰 폭으로 소모전력을 줄이기 위한 연구를 진행할 계획이다.

REFERENCES

[1] A. Sodani, and Processor, C. A. M, “Race to Exascale: Opportunities and Challenges,” In MICRO 2011 Keynote, 2011.

[2] NVIDIA Tegra 4 Family CPU Architecture, NVIDIA, Tech. Rep., 2013. [Online]. Available: http://www.nvidia.com/docs/IO/116757/NVIDIA_Quad_a15_whitepaper_FINALv2.pdf”.

[3] A. Sembrant, E. Hagersten, and D. Black-Shaffer, “TLC: A Tag Less Cache for Reducing Dynamic First Level Cache Energy,” in Proc. of IEEE/ACM International Symposium on Microarchitecture, pp. 49-61, 2013.

[4] M. D. Powell, A. Agarwal, T. vijaykumar, B. Falsafi, and K. Roy, “Reducing Set-associative Cache Energy via Way-Prediction and Selective Direct-mapping,” in MICRO, pp. 54-65, 2001.

[5] W. Zhang, H. Zhang, and J. Lach, “Reducing Dynamic Energy of Set-associative L1 Instruction Cache by Early Tag Lookup,”, Low Power Electronics and Design, pp.49-54, 2015.

[6] J. Dai, M. Guan, and L. Wang, “Exploiting Early Tag Access for Reducing L1 data cache energy in embedded processors,” IEEE Transactions on Very Large Scale Integration Systems, Vol. 22, NO. 2, pp.396-407, 2014.

[7] C. Zhang, F. Vahid, J. yang, and W. najjar, “A Way-Halting Cache for Low-Energy High-Performance Systems,” ACM Transactions on Architecture and Code optimization, Vol. 2, No. 1, pp.34-54, 2005.

[8] J. Dai, and L. Wang, “An Energy-Efficient L2 Cache Architecture using Way Tag Information under Write-through Policy,” IEEE Transactions on Very Large Scale Integration Systems, Vol.21, No. 1, pp. 102-112, 2013.

[9] D. Sanchez, and C. Kozyrakis, "The ZCache: Decoupling Ways and Associativity," In Microarchitecture, pp. 187-198, 2010.

[10] A. Seznec, "A Case for Two-Way Skewed-Associative Caches," In ACM SIGARCH Computer Architecture News, Vol.21, No. 2, pp. 169-178, 1993.
[11] A. Seznec, and F. Bodin, "Skewed-Associative Caches," Parallel Architectures and Languages Europe, pp. 305-316, 1993.
[12] C.L Yang, and C. L., "Hotspot Cache: Joint Temporal and Spatial Locality Exploitation for I-cache Energy Reduction," Low Power Electronics and Design, pp. 114-119, 2004.
[13] J. Ye, H. Ding, Y. Hu, and T. Watanabe, "A Behavior-based Adaptive Access-Mode for Low-Power Set-Associative Caches in Embedded systems," Jornal of Information processing, Vol.20, No. 1, pp. 26-36, 2012.
[14] A. ma, M. Zhang and K. Asanovic, "Way Memorization to Reduce Fetch Energy in Instruction Caches," ISCA Workshop on Complexity Effective Design, Vol.20, pp. 31, 2001.
[15] C. H. Kim, S. W. Chung, and C. S Jhon, "A Power-aware Branch Predictor by Accessing BTB Selectively," Jornal of Computer Science and Technology, Vol.20, No.5, pp. 607-614, 2005.
[16] T. Austin, E., Larson, and D. Ernst, "SimpleScalar: An Infrastructure for Computer System Modeling," Computer, Vol.35, No.2, pp. 59-67, 2002.
[17] Wattch, http://www.eecs.harvard.edu/~dbrooks/
[18] SPEC Benchmark Suite. Information available at http://spec.org/cpu2006/
[19] SPEC CPU2000 Benchmarks, http://www.specbench.org
[20] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "CACTI 6.0: A Tool to Model Large Caches," Technical Report HPL-2009-85, Hewlett Packard Laboratories, 2009.

Authors

Gwang Bok Kim received the B.S degree and M.S in electronics and computer engineering from Chonnam National University, Gwangju, Korea in 2013 and 2015 respectively.
He is currently a Ph.D student at Chonnam National University. His research interests include computer architecture, low power systems, and GPGPU.

Jong Myon Kim received the B.S. degree in electrical engineering from the Myong-Ji University, Yong-In, Korea, in 1995, the MS degree in electrical and computer engineering from University of Florida, Gainesville, in 2000, and the Ph.D. degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, in 2005.
He was a senior research staff in the Chip Solution Center of Samsung Advanced Institute of Technology from 2005 to 2007. Since 2007, he has been with the School of Computer Engineering and Information Technology at the University of Ulsan, Ulsan, Korea, where he is currently an Associate Professor. His research interests include embedded systems, application-specific processors, and parallel processing.

Cheol Hong Kim received the B.S. degree in Computer Engineering from Seoul National University, Seoul, Korea in 1998 and M.S. degree in 2000. He received the Ph.D. in Electrical and Computer Engineering from Seoul National University in 2006.
He worked as a senior engineer for SoC Laboratory in Samsung Electronics, Korea from Dec. 2005 to Jan. 2007. Now is working as an Associate Professor at School of Electronics and Computer Engineering, Chonnam National University, Korea. His research interests include embedded systems, mobile systems, computer architecture, SoC design, low power systems, and multiprocessors.