내장형 프로세서를 위한 저전력 분기 예측기 설계 기법

김 철 홍[†] · 송 성 근[††]

요 약

프로세서의 파이프라인 길이가 점차 길어지고 한 사이클에 이슈되는 명령어의 수가 증가함에 따라, 분기 예측기의 정확도는 프로세서의 성능에 상당한 영향을 미치게 되었다. 또한, 내장형 프로세서를 설계하는데 있어서는 전력 효율성이 가장 중요한 설계 고려 사항 중 하나가 되었다. 그러므로, 내장형 프로세서의 분기 예측기를 설계할 때에는 성능과 전력 효율성이 함께 고려되어야 한다. 본 논문에서는 gshare 분기 예측기가 적용된 내장형 프로세서에서 선택적인 BTB (Branch Target Buffer) 접근을 가능하게 하는 저전력 분기 예측기를 제안하고자 한다. 제안하는 분기 예측기 내에서 BTB는 직전 명령어가 테이큰 (Taken) 분기로 예측되지 않는 경우에는, PHT (Pattern History Table)의 예측 결과가 테이큰인 경우에만 접근된다. PHT의 예측 결과가 테이큰인 분기 명령어의 경우에만 다음에 인출될 명령어의 주소를 BTB 접근을 통해 얻은 주소로 결정하기 때문이다. 물론, 이와 같은 선택적인 BTB 접근으로 인하여 성능 저하가 발생하는 것을 방지하기 위해 직전 명령어가 테이큰 분기로 예측된 경우에는 PHT의 예측 결과에 관계없이 BTB는 항상 접근된다. 선택적인 BTB 접근을 하기 위해, 제안하는 분기 예측기 내의 PHT는 기존 분기 예측기의 PHT와 비교하여 1 사이클 일찍 접근되도록 구현한다. 1 사이클 빠른 접근을 위해 제안하는 PHT는 한 번의 접근을 통해 두 개의 예측 결과를 동시에 얻어오게 구현하고, 이를 통해 PHT의 접근 횟수도 줄임으로써 분기 예측기의 전력 소모를 줄이는 효과 또한 얻게 된다. 제안하는 분기 예측기는 하드웨어 오버헤드나 예측 정확도의 감소 없이 전력 소모를 줄일 수 있다는 장점을 가진다. 실험 결과에 따르면, 제안하는 분기 예측기는 기존의 분기 예측기와 비교하여 35~48%의 전력 소모를 줄이는 결과를 보인다.

키워드 : 분기 예측기, BTB, PHT, 내장형 프로세서, 저전력 시스템

A Power-aware Branch Predictor for Embedded Processors

Cheol Hong Kim[†] · Sung-Gun Song[††]

ABSTRACT

In designing a branch predictor, in addition to accuracy, microarchitects should consider power consumption, especially for embedded processors. This paper proposes a power-aware branch predictor, which is based on the gshare predictor, by accessing the BTB (Branch Target Buffer) only when the prediction from the PHT (Pattern History Table) is taken. To enable the selective access to the BTB, the PHT in the proposed branch predictor is accessed one cycle earlier than the traditional PHT to prevent the additional delay. As a side effect, two predictions from the PHT are obtained through one access to the PHT, which leads to more power savings. The proposed branch predictor reduces the power consumption, not requiring any additional storage arrays, not incurring additional delay (except just one MUX delay) and never harming accuracy. Simulation results show that the proposed predictor reduces the power consumption by 35~48 % compared to the traditional predictor.

Key Words : Branch predictor, BTB, PHT, Embedded processor, Low power system

1. 서 론

프로세서의 파이프라인 길이(Pipeline Depth)가 길어지고 한 사이클당 이슈(Issue) 되는 명령어의 수가 증가하면서, 분기 예측 (Branch Prediction) 실패로 인한 프로세서의 성능 저하 정도가 점차 커지게 되었다. 이로 인해, 최근의 내장형 프로세서 (Embedded Processor)의 성능은 분기 예측기 (Branch Predictor)의 정확도에 큰 영향을 받게 되었다. 이러한 이유로, 내장형 프로세서를 설계하는데 있어서 분기 예측기의 정확성을 높이는 기법은 설계 시 중요한 고려 사항이 되고 있다. 또한, 프로세서의 전력 효율성이 칩의 경쟁력에 미치는 영향이 점차 커짐에 따라, 내장형 프로세서의 분기 예측기를 설계할 때에는 정확성과 함께 전력 효율성도 반드시 고려되어야 한다.

일반적으로 분기 예측기는 크게 두 개의 하드웨어 모듈로 구성된다. 하나는 분기 방향 예측기 (Branch Direction Predictor)이고 다른 하나는 분기 목적 주소 예측기 (Branch Target Predictor)이다. PHT (Pattern History Table)로 구현되는 분기 방향 예측기는 분기 명령어 (Branch Instruction)

※ 이 논문은 2007년도 전남대학교 학술연구비 지원에 의하여 연구되었음.
† 정 회 원 : 전남대학교 전자컴퓨터공학부 전임강사
†† 준 회 원 : 전남대학교 전자컴퓨터공학부 박사과정
논문접수 : 2007년 8월 13일, 심사완료 : 2007년 10월 8일

의 테이큰 (Taken) 또는 언테이큰 (Untaken) 여부를 예측한다. BTB (Branch Target Buffer)로 구현되는 분기 목적 주소 예측기는 테이큰 분기 명령어의 목적 주소 (Target Address)를 예측한다. 이하 본 논문에서의 분기 예측기는 PHT와 BTB로 구성되는 것으로 가정한다.

PHT와 BTB는 상당히 큰 크기를 가지고 있고 명령어가 인출(Fetch)될 때마다 접근이 이루어지기 때문에 상당히 많은 전력을 소모하게 된다. 일부 범용 프로세서에서는 선디코딩(Predecoding) 기법을 통해 인출된 명령어가 분기 명령어인지를 미리 검사하여, 분기 명령어인 경우에만 선택적으로 분기 예측기(PHT와 BTB)를 접근함으로써 전력 소모를 줄이고 있다. 하지만, 이를 위해서는 분기 예측기를 접근하기에 앞서 미리 명령어 캐쉬(Instruction Cache)를 접근하여야 한다. 이 경우, 만약 파이프라인에서 인출 단계(Fetch Stage)가 timing-critical한 단계라면, 명령어 캐쉬와 분기 예측기의 순차적 접근으로 인하여 인출 단계의 시간 지연이 커지면서 프로세서의 성능이 크게 저하될 것이다. 그러므로, ARM 1136[1] 이나 ARM 1156[2]과 같은 내장형 프로세서에서는 timing-critical한 부분인 인출 단계의 지연 시간을 줄이기 위해 분기 예측기와 명령어 캐쉬가 동시에 접근된다. 즉, 명령어 캐쉬와 분기 예측기의 동시 접근을 위해 내장형 프로세서의 분기 예측기는 명령어가 인출될 때마다 접근되어야 하므로 상당히 많은 전력을 소모하는 결과를 가져온다. 최근 연구 결과에 따르면, BTB (Branch Target Buffer)를 포함한 분기 예측기의 전력 소모량은 전체 프로세서 전력 소모량의 10%를 넘는다고 한다[3]. 그러므로, 분기 예측기의 전력 효율성을 높이는 것은 프로세서의 전력 효율성을 높이는 데 있어서 큰 역할을 담당할 것으로 기대된다.

본 논문에서는 내장형 프로세서의 분기 예측기를 설계하는데 있어서 성능 저하 없이 전력 소모를 줄일 수 있는 하드웨어 기법을 제안하고자 한다. 본 논문의 연구 대상은 분기 예측기와 명령어 캐쉬가 인출 단계에서 동시에 접근되는 내장형 프로세서이다. 제안하는 분기 예측기는 BTB 접근으로 인한 전력 소모를 줄이기 위해, 직전 명령어가 테이큰 분기로 예측되지 않는 경우에는 PHT의 예측 결과가 테이큰인 경우에만 BTB를 접근한다. 직전 명령어가 테이큰 분기로 예측되는 경우에는 제안하는 구조로 인해 성능 저하가 생기는 것을 방지하기 위해 BTB를 조건에 관계없이 항상 접근한다. 이와 같은 선택적인 BTB 접근을 성능 저하 없이 구현하기 위해, 제안하는 분기 예측기의 PHT는 기존 분기 예측기의 PHT에 비해 1 사이클 일찍 접근된다. 물론, 직전에 수행된 분기 명령어로 인해 글로벌 히스토리(Global History)가 변경된 경우에는 기존의 PHT와 같은 사이클에 접근이 되도록 한다. 1 사이클 빠른 접근을 위해 제안하는 PHT는 한 번의 접근으로 두 개의 예측 결과를 가져올 수 있도록 구현한다. 이를 통해 제안하는 분기 예측기는 한 번의 PHT 접근을 통해 두 개의 예측 결과를 가져오게 되므로 PHT의 접근 횟수 또한 감소하게 되므로 BTB 뿐만 아니라 PHT에서의 전력 소모도 감소할 것으로 기대된다.

이하 본 논문의 구성은 다음과 같다. 2장에서는 분기 예측기의 전력 소모를 줄이기 위한 기존의 연구들을 살펴본다. 3장에서는 기존의 분기 예측기를 설명하고, 4장에서는 제안하는 저전력 분기 예측기의 구조를 기술한다. 5장에서는 정성적인 분석과 함께 전력 소모량을 측정하기 위한 실험 방법과 실험 결과를 보여주고, 6장에서 결론을 맺는다.

2. 관련 연구

최근에는 프로세서의 전력 효율성이 설계 시 중요한 고려사항이 되면서, 분기 예측에서 소모되는 전력을 줄이기 위한 연구도 많이 이루어졌다. 파이프라인 게이팅(Pipeline Gating)은 분기 명령어에 대한 예측 결과의 정확도를 미리 예상하는 기법이다[4]. 이 기법에서는 분기 명령어에 대한 예측 정확도가 낮을 것으로 예상되는 경우에는, 분기 예측 실패(Branch Misprediction)로 인한 전력 소모를 줄이기 위해 미리 명령어 인출을 중단한다. 이 기법은 본 논문에서 제안하는 기법과 달리 분기 예측 실패로 인한 전력 소모를 줄이기 위한 방법일 뿐, 분기 예측기 자체의 전력 효율성을 높이기 위한 기법은 아니다. 분기 예측기(PHT, BTB)의 모든 엔트리(Entry)가 항상 사용되고 있지는 않는다는 점에 착안하여 캐쉬에 적용한 Decay 기법을 분기 예측기에 적용함으로써 정적 전력 소모량(Leakage Power Dissipation)을 줄이기 위한 기법도 제안되었다[5]. 이 연구와 달리 본 논문에서 제안하는 기법은 정적 전력 소모량이 아닌 동적 전력 소모량 (Dynamic Power Dissipation)에 초점을 맞추고 있다.

ACBTB (Application Customizable BTB)는 BTB에 대한 접근 횟수 감소를 통해 분기 예측에 따른 전력 소모를 줄이는 기법이다[6]. 이 기법은 컴파일 시간에 프로그램에 대한 정보를 추출하고, 이 정보를 바탕으로 인출되는 명령어가 분기 명령어가 아닌 경우에는 BTB를 접근하지 않는다. 컴파일러를 이용하여 특정 명령어를 미리 삽입함으로써 분기 예측기에 대한 접근을 줄이는 방법 또한 제안되었다[7][8]. 이들 기법과 달리 본 논문에서 제안하는 기법은 컴파일러의 도움 없이 하드웨어만으로 구현된다.

Profiling을 통해 수집된 정보를 이용하여 분기 예측에 필요한 하드웨어 자원을 동적으로 관리하는 기법도 제안되었다[9][10]. 이 기법은 수집된 정보를 기반으로 혼합 분기 방향 예측기 (Hybrid Branch Direction Predictor)의 일부 요소를 비활성화시키거나 BTB의 엔트리 수를 상황에 맞게 동적으로 조절하는 기법을 통해 전력 소모량을 줄여준다. 이 기법은 복잡한 하드웨어로 구성되는 고성능 프로세서의 혼합 분기 예측기에만 적용 가능한 기법이다. 본 논문에서 제안하는 기법은 이와 달리 간단한 구조의 분기 예측기가 사용되는 내장형 프로세서를 목표로 하고 있다.

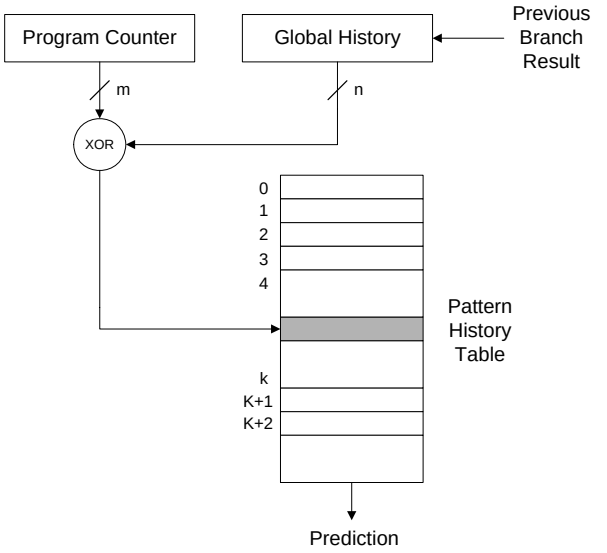
뱅킹 (Banking) 기법 또한 분기 예측기의 전력 소모를 줄일 수 있는 기법이다. 여러 뱅크들로 분기 예측기 테이블을 구현하고, 분기 예측기 접근 시 하나의 뱅크만 활성화함으

로써 분기 예측기 접근에 필요한 시간 및 전력 소모를 줄일 수 있다. 하지만, 뱅킹 기법을 통해 얻을 수 있는 전력 소모 감소 효과는 4%를 넘지 않고, 뱅킹을 구현하기 위해서는 부가적인 칩 공간 (Chip Area)이 필요하게 된다[3][11]. PPD (Prediction Probe Detector) 모듈은 명령어 캐쉬 접근 전에 접근이 이루어지는 테이블로 저전력 분기 예측기를 위해 제안되었다[3][11]. PPD는 명령어 캐쉬와 같은 수의 엔트리를 가지는 하나의 테이블이고, 각각의 엔트리는 2비트로 구성된다. 하나의 비트는 PHT 의 선택적인 접근을 위한 것이고, 다른 비트는 BTB의 선택적인 접근을 위해 사용된다. 하지만, 이 기법은 PPD 자체가 별도의 전력과 칩 공간을 소모한다는 단점을 가진다. 또한, 이 기법은 파이프라인 지연 시간 (Pipeline Latency)를 증가시켜, 전체 프로세서의 성능을 저하시키는 결과도 가져온다[12][13]. 본 논문에서 제안하는 분기 예측기는 별도의 칩 공간을 거의 차지하지 않고, 구현 시 하나의 MUX delay만 추가된다는 장점을 가진다. 이와 함께, 분기 예측기의 정확성에는 전혀 영향을 미치지 않는다는 장점도 가지고 있다.

기존에 제안된 저전력 분기 예측기에 관한 연구는 거의 복잡도가 높은 분기 예측기들을 대상으로 이루어졌다. Bimodal, gshare 분기 예측기와 같이 상대적으로 복잡도가 낮은 분기 예측기에 대해서는 저전력 연구가 거의 이루어지지 않았다. 본 논문에서는, 향후 임베디드 프로세서에서 많이 사용될 것으로 예측되는 gshare 분기 예측기를 대상으로 저전력 기법을 제안함으로써 임베디드 프로세서의 전력 효율성을 더욱 높이고자 한다.

3. 내장형 프로세서의 일반적인 분기 예측기

내장형 프로세서에는 프로그램 내 각각의 분기 명령어에 대해 고정된 예측을 수행하는 정적 분기 예측기(Static Branch Predictor)가 많이 사용되어 왔다. 몇몇 내장형 프로세서들에는 정적 분기 예측기보다 높은 정확성을 가지는 bimodal 분기 예측기가 사용되었다. 하지만, 프로세서의 파이프라인 길이가 길어지고 한 사이클에 이슈되는 명령어의 수가 점차 많아지면서 분기 예측 실패로 인한 성능 저하 문제가 심각해지게 되었다. 그 결과, 내장형 프로세서에도 앞서 사용된 분기 예측기보다 높은 정확성을 가지는 분기 예측기가 필요하게 되었다. 단지 정확성만을 따진다면, Alpha21264에 적용되는 토너먼트(tournament) 분기 예측기 [14]와 같은 복잡한 구조의 분기 예측기를 사용하는 것이 좋은 선택일 것이다. 하지만, 토너먼트 분기 예측기와 같이 정확성이 높은 분기 예측기는 내장형 프로세서에 적용하기에는 너무 많은 칩 공간을 차지한다. 이러한 이유로, 최근에는 적당한 칩 공간과 전력을 소모하면서 비교적 높은 정확성을 제공하는 gshare[14]와 같은 분기 예측기가 내장형 프로세서에 적용 검토되고 있다[2]. 본 논문에서는 향후 사용될 내장형 프로세서를 위한 분기 예측기의 제안을 위해 gshare 분기 예측기를 사용하는 내장형 프로세서를 목표 대상으로 한다.

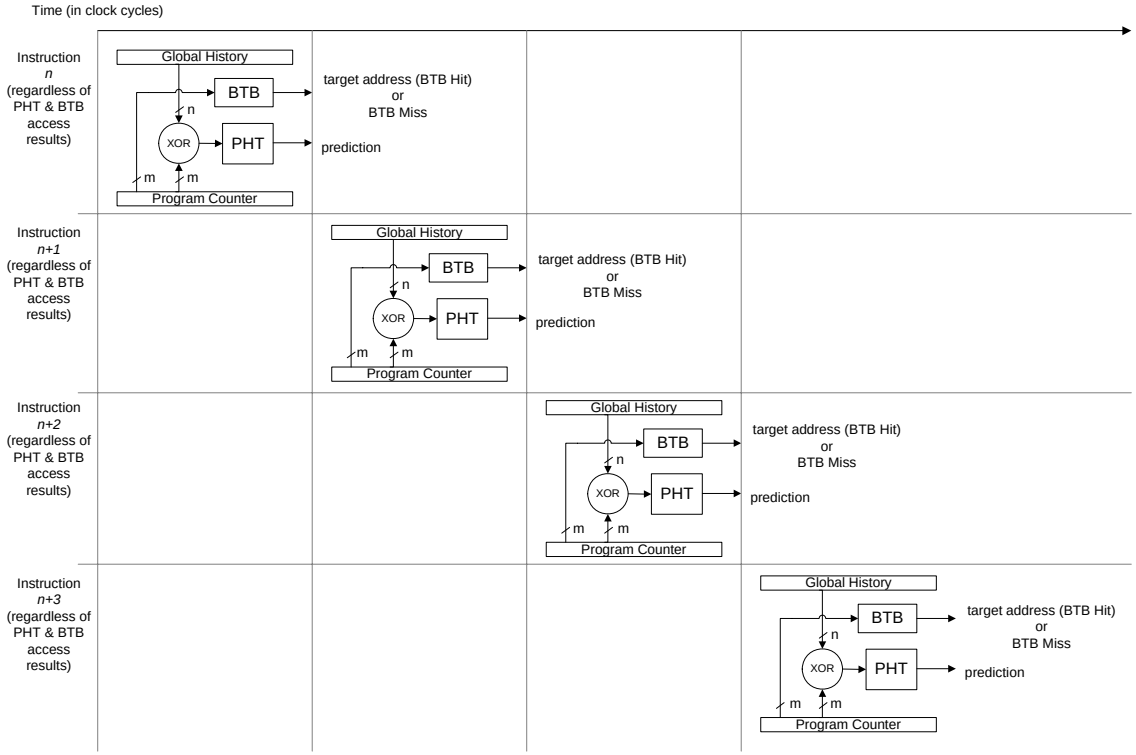


(그림 1) 저전력 기법을 적용하지 않은 기존 gshare 분기 예측기의 PHT 구조

그림 1은 저전력 기법을 적용하지 않은 기존 gshare 분기 예측기의 PHT 구조를 보이고 있다. Gshare 분기 예측기의 PHT는 2비트 카운터 (Counter)의 배열로 구현된다. PC (Program Counter) 값과 글로벌 히스토리 (Global History)를 exclusive OR하여 인덱싱 (Indexing)되는 PHT 각 엔트리의 카운터는 대응된 분기 명령어가 테이큰되면 1 증가하고, 언테이큰되면 1 감소한다. 이를 통해, PHT를 접근하는 명령어에 대한 분기 예측은 각 엔트리 카운터의 MSB (Most Significant Bit) 값에 따라 결정된다 (1이면 테이큰, 0이면 언테이큰).

기존의 분기 예측기에서는 (그림 2)에서 보이는 바와 같이 인출 단계에서 PHT와 BTB가 명령어 캐쉬와 동시에 접근되어야 한다. 명령어를 캐쉬로부터 읽어오기 전에는 인출하는 명령어가 분기 명령어인지 여부를 판단할 수 없기 때문이다. 즉, PHT와 BTB는 인출되는 명령어가 분기 명령어인지 여부에 관계없이 항상 읽기 접근이 이루어져야 하는 것이다. 읽기 접근 후, 인출된 명령어가 분기 명령어로 판정되면 PHT는 추후에 엔트리 카운터의 값을 업데이트하기 위해 쓰기 접근 되어야 한다. 인출된 명령어가 분기 명령어가 아닌 것으로 판정되면, PHT에 대한 쓰기 접근을 할 필요는 없다. BTB의 경우에는 읽기 접근 후, BTB를 통해 예측된 목적 주소(Target Address)가 틀린 것으로 판정되는 경우 (BTB Misprediction)에 정보의 변경을 위해 쓰기 접근된다.

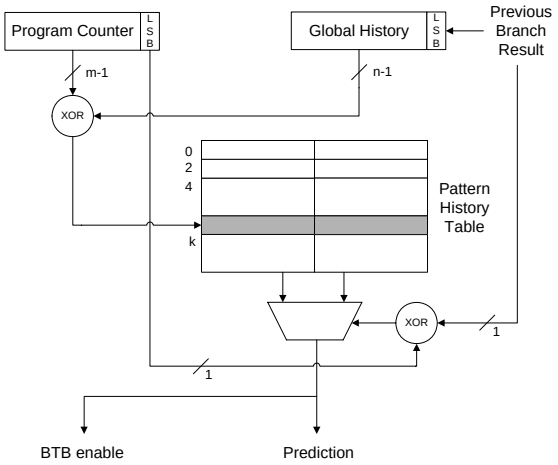
본 논문에서 목표로 하는 인출 단계에 시간적 여유가 없는 프로세서들에서는 BTB 접근이 히트(Hit)인 경우에는 해당 명령어를 분기 명령어일 것으로 예측한다. 그러므로, BTB에 대한 접근이 히트이고 PHT를 통한 예측이 테이큰이면(테이큰 분기) 다음 명령어는 BTB를 통해 얻은 목적 주소를 이용하여 인출한다. 그렇지 않은 경우에는 주소상으로 현재 인출된 명령어의 바로 다음 명령어를 인출한다.



(그림 2) 기존 분기 예측기의 분기 예측

4. 저전력 분기 예측기

앞 장에서 살펴본 바와 같이 기존 분기 예측기는 명령어가 인출될 때마다 PHT와 BTB에 대한 접근을 한다. 이와 달리, 제안하는 분기 예측기는 직전 명령어(바로 직전 사이클에 인출된 명령어)가 테이큰 분기 명령어로 예측되지 않는 경우에는 PHT의 예측 결과가 테이큰인 경우에만 BTB를 접근하고자 한다. 성능 저하 없이 BTB에 대한 선택적인 접근을 하기 위해서는, PHT 접근이 BTB 접근보다 먼저 이루어져야 하므로 제안하는 분기 예측기의 PHT는 기존 분기 예측기의 PHT에 비해 1 사이클 먼저 접근된다.

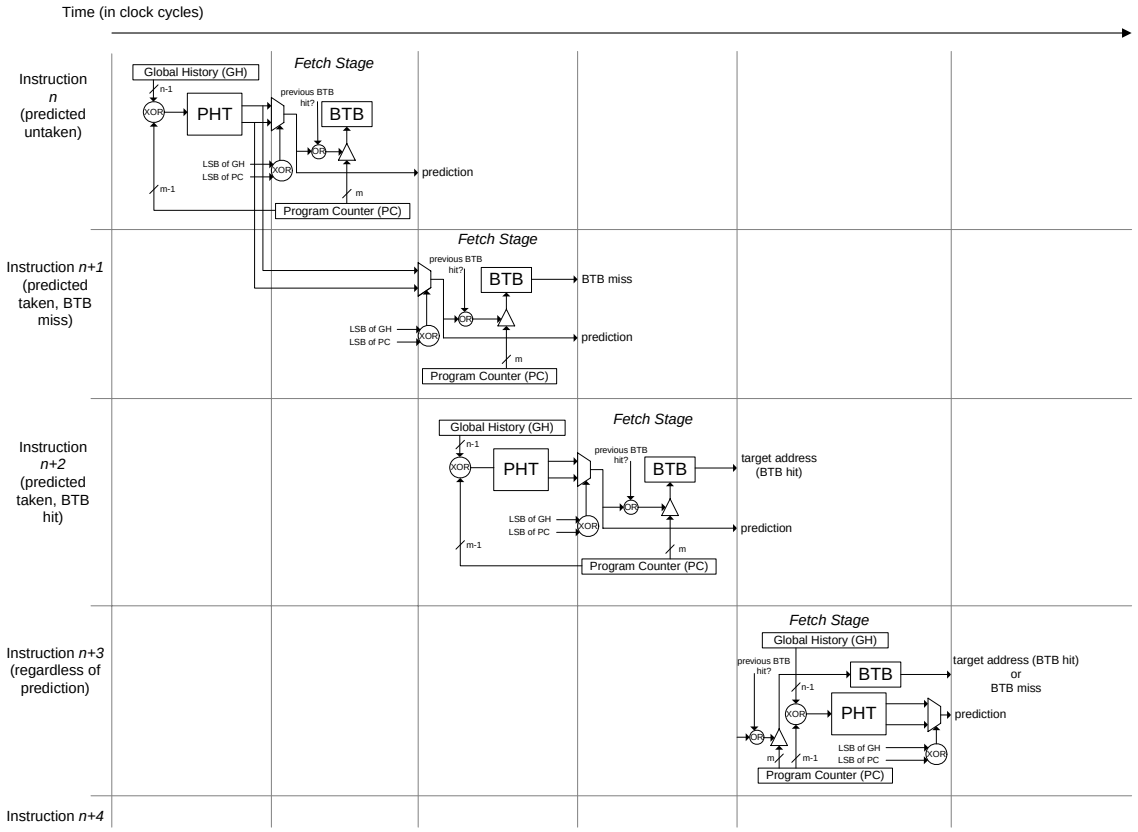


(그림 3) 제안하는 gshare 분기 예측기의 PHT 구조

4.1 PHT에 대한 이른 접근

기존 분기 예측기의 PHT는 명령어가 인출될 때마다 항상 접근된다. 선택적인 BTB 접근을 위해 필요한 1 사이클 이른 PHT 접근과 PHT에서의 소모 전력을 줄이기 위해, 제안하는 PHT는 한 번의 읽기 접근을 통해 두 개의 예측 결과를 얻어오도록 구현한다. 제안하는 PHT 구조는 그림 3과 같다. 그림에서 보는 바와 같이, 제안하는 구조에서는 PHT의 폭 (Width)을 기존 PHT에 비해 두 배로 늘리는 대신에, 공정한 비교를 위해 PHT의 깊이 (Depth)를 절반으로 줄인다. 한 번 접근 시, 제안하는 PHT는 기존의 PHT와 비교하여 거의 유사한 양의 전력을 소모한다 (삼성 메모리 컴파일러(Samsung Memory Compiler[15])를 통한 측정 결과, 2048×4 크기의 PHT는 4096×2 크기의 PHT에 비해 한 번 접근 시 4%의 전력만을 더 소모한다).

기존 분기 예측기의 PHT와 달리, 제안하는 분기 예측기의 PHT는 PC 값에서 LSB(Least Significant Bit)를 제외한 값과 글로벌 히스토리에서 LSB를 제외한 값을 exclusive OR 연산하여 해당 엔트리를 인덱싱한다. 이 경우, 글로벌 히스토리의 값이 바뀌지 않고(분기 명령어로 인한 업데이트가 발생하지 않고), PC의 LSB 값만 바뀐다면 한 번의 PHT 읽기 접근을 통해 연속적인 두 개의 명령어에 대한 예측 결과를 얻을 수 있다. 기본적으로, 제안하는 PHT는 직전 명령어가 PC 값과 글로벌 히스토리를 변경시키는 테이큰



(그림 4) 제안하는 분기 예측기의 분기 예측

분기 명령어가 아닐 것을 가정한다. 다시 말해, PHT에 대한 연속적인 접근을 가정하는 것이다. 만약에 직전 명령어가 테이큰 분기 명령어로 예측된다면(직전 명령어에 대한 BTB 접근이 히트라면) 제안하는 PHT는 새로운 PC값과 새로운 글로벌 히스토리를 이용하여 재접근된다.

그림 4에서 보이는 바와 같이 직전 명령어가 테이큰 분기 명령어로 예측되지 않는 경우에 제안하는 PHT는 기존의 PHT에 비해 1 사이클 일찍 접근된다. 빠른 접근을 통해 얻어온 두 개의 예측 결과 중에서 현재 명령어에 대한 예측 결과로는 인출 단계에서 MUX를 사용하여 PC의 LSB와 글로벌 히스토리의 LSB를 exclusive OR 함으로써 둘 중 하나가 선택된다. 그러므로, 기존 분기 예측기와 비교하여 비교적 빠른 시기인 하나의 MUX 지연시간 후에 PHT의 예측 결과를 사용할 수 있게 되는 것이다(그림 4의 명령어 n, 명령어 n+1, 명령어 n+2의 경우). 만약 직전 명령어(그림 4의 명령어 n+2)가 테이큰 분기 명령어로 예측된다면, 현재 인출되는 명령어(그림 4의 명령어 n+3)에 대한 PHT 접근은 새로운 PC 값과 글로벌 히스토리를 사용하여 인출 단계에서 새롭게 재접근되어야 한다. 물론, 직전 명령어가 테이큰 분기 명령어가 아니더라도 분기 명령어로 인해 글로벌 히스토리의 내용이 변경된다면 PHT는 새롭게 접근되어야 한다.

제안하는 PHT 내에서 한 번의 접근을 통해 얻은 두 개의 예측 결과를 모두 사용할 수 있는 경우는 명령어가 순차적으로 사용되는 경우에만 가능하다. 즉, (그림 4)의 명령어 n과 명령어 n+1과 같이 두 개의 PHT 예측 결과를 한 번의 PHT 접근으로 얻을 수 있는 경우는 두 명령어가 순차적으로 PHT에 할당되었을 경우에만 가능하다는 것이다. 만약 PHT가 항상 순차적으로만 접근된다면, 제안하는 PHT는 기존의 PHT와 비교하여 50%의 접근 횟수를 줄일 수도 있다. 하지만, 실제적으로는 분기 명령어들로 인해 항상 순차적인 접근이 이루어지지는 않게 된다. 분기 명령어는 테이큰 여부에 관계없이 글로벌 히스토리의 내용을 변경시키고, 테이큰 분기 명령어는 PC 값도 변경시키기 때문이다. 그러므로, 제안하는 PHT에 대한 접근 횟수는 최대 ((기존의 PHT에 대한 접근 횟수)÷2 + 분기 명령어의 수)까지 줄어들게 될 것이다. 일반적으로 대부분의 응용 프로그램[16][17]에서 분기 명령어는 전체 명령어의 0~30% 정도를 차지한다. 이러한 점을 감안하면 제안하는 PHT는 기존 PHT와 비교하여 접근 횟수를 상당히 줄임으로써 전력 소모 감소 효과를 얻을 수 있을 것으로 기대된다.

위에서 설명한 바와 같이, 제안하는 분기 예측기의 PHT는 기존 분기 예측기의 PHT와 비교하여 상당량의 전력 소모를 줄일 수 있을 것이다. 그렇다면, 분기 예측기의 정확도에는 어떤 영향을 미치는지 생각해 보자. 만약 직전 명령어가 테이큰 분기로 예측되지 않고 글로벌 히스토리도 변경되지 않았다면 한 번의 PHT 접근을 통해 얻은 두 개의 예측 결과는 기존 PHT에 대한 두 번의 접근으로 얻은 결과와 같게 되므로 정확성에 있어서 영향을 받지 않는다. 만약 직전

명령어가 테이큰 분기 명령어로 예측되거나 분기 명령어로 인해 글로벌 히스토리가 변경된 경우에는, 제안하는 PHT는 새로운 PC 값과 글로벌 히스토리를 이용하여 새롭게 재접근된다. 이 경우, 첫 번째 PHT 접근을 통해 얻은 예측 결과는 기존 PHT와 비교하여 다르지만 재접근을 통해 얻은 예측 결과는 기존 PHT의 접근을 통해 얻은 예측 결과와 같게 된다. 그러므로, 어떠한 경우에도 제안하는 분기 예측기의 정확성은 기존 분기 예측기의 정확성과 비교하여 달라지지 않는다.

4.2 BTB에 대한 선택적인 접근

기존 분기 예측기의 BTB는 PHT와 마찬가지로 명령어가 인출될 때마다 접근되어야 한다. 하지만, 제안하는 분기 예측기는 PHT의 예측 결과를 인출 단계의 시작 부분 (1 MUX 지연 시간 후)에 알 수 있게 되므로, BTB에 대한 선택적인 접근을 가능하게 한다(그림 4). 제안하는 분기 예측기에서는 PHT의 예측 결과가 테이큰인 경우에만 BTB를 접근한다(그림 4의 명령어 n+1과 n+2의 경우). PHT의 예측 결과가 언테이큰인 경우에는 BTB 접근을 통해 얻어지는 목적 주소가 사용되지 않을 것이므로 BTB에 대한 접근이 이루어지지 않는다(그림 4의 명령어 n). 만약 직전 명령어가 테이큰 분기 명령어로 예측되는 경우에는 앞 절에서 설명한 바와 같이 PHT를 재접근해야 하므로, PHT와 BTB에 동시 접근이 이루어지기 때문에 BTB는 항상 접근된다(그림 4의 명령어 n+3).

일반적으로, PHT에서 언테이큰으로 예측되는 명령어는 전체 명령어의 10~60%를 차지한다[16][17]. 그러므로, 직전 명령어가 테이큰 분기 명령어로 예측되지 않는 경우에 있어서 제안하는 분기 예측기는 BTB에 대한 접근 횟수를 10~60% 정도 줄임으로써 기존의 분기 예측기와 비교하여 상당량의 전력 소모를 절감할 수 있을 것으로 기대된다.

5. 모의 실험 방법

이 장에서는 분기 예측기 구조에 대한 전력 소모 분석 모델을 제시하고 이를 사용하여 정성적인 분석을 한다. 또한, 모의 실험을 통해 기존 분기 예측기와 제안하는 분기 예측기의 전력 소모량을 비교 분석한다. 제안하는 분기 예측기의 성능 및 전력 효율성을 측정하기 위해 비교 대상으로 저전력 기법을 적용하지 않은 기존의 분기 예측기를 사용한 이유는 기존의 연구를 통해 제안된 저전력 분기 예측기 기법들은 주로 gshare 분기 예측기보다 훨씬 복잡한 분기 예측기들을 대상으로 이루어졌기 때문이다. Gshare 분기 예측기와 같이 구성이 단순한 분기 예측기에 대해서는 기존에는 저전력 관련 연구가 거의 이루어지지 않았다.

모의 실험은 SimpleScalar 시뮬레이터를 수정하여 수행하였다[18]. 실험에 사용한 전력 소모 관련 변수들은 25℃, Vdd = 1.20V 환경에서 0.13 um 일반 공정을 가정하여 삼성 메모리 컴파일러를 통해 얻은 값을 사용하였다 [15]. 실험에 사용한 프로세서 모델은 ARM 1136 [1] 명세서를 기반으로 구성하였다. SimpleScalar 시뮬레이터의 입력으로는 SPEC CPU2000 벤치마크 프로그램들을 사용하였다 [16].

<표 1>은 실험에 사용한 PHT와 BTB의 크기 및 읽기 쓰기 시 소모되는 전력을 보여준다. PHT와 BTB의 크기는 향후 내장형 프로세서에서 사용될 것으로 예상되는 크기로 설정하였고, 표에 나타낸 전력 소모량은 4096×2 크기의 PHT를 한 번 읽는 데 소모되는 전력량으로 정규화한 값을 보여주고 있다.

〈표 1〉 PHT와 BTB의 크기에 따른 읽기/쓰기 시 소모 전력

	적용 대상	크 기	읽 기	쓰 기
PHT	기존 분기 예측기	4096 × 2	1.00	0.91
PHT	제안하는 분기 예측기	2048 × 4	1.04	0.99
BTB	기존/제안하는 분기 예측기	256 entries	1.88	2.14

〈표 2〉 분석 모델에 사용되는 기호와 의미

기 호	의 미
P(trad.)	기존 분기 예측기의 총 전력 소모량
P(prop.)	제안하는 분기 예측기의 총 전력 소모량
Ppht_traditional_read	기존 PHT의 한 번 읽기 접근 시 전력 소모량
Ppht_traditional_write	기존 PHT의 한 번 쓰기 접근 시 전력 소모량
Ppht_proposed_read	제안하는 PHT의 한 번 읽기 접근 시 전력 소모량
Ppht_proposed_write	제안하는 PHT의 한 번 쓰기 접근 시 전력 소모량
Pbtb_read	BTB의 한 번 읽기 접근 시 전력 소모량
Pbtb_write	BTB의 한 번 쓰기 접근 시 전력 소모량
Ninst	전체 명령어 수
Nbranch	분기 명령어의 수
Nbtb_mispredictions	분기 명령어에 대한 BTB 예측 실패 횟수
Nuntaken_predictions	제안하는 PHT에서 언테이큰으로 예측되는 명령어의 수
Nuntaken_previous_btb_hit	직전 명령어가 테이큰 분기 명령어로 예측되는 경우의 PHT 예측이 언테이큰인 명령어의 수

5.1 정성적 분석 결과

모의 실험 결과를 보이기에 앞서 분석 모델을 통한 정성적인 분석을 보인다. 분기 예측기의 전력 소모량을 측정하기 위한 분석 모델에 사용되는 기호는 <표 2>에 기술하였다.

기존 분기 예측기에서 소모되는 전력의 총합은 다음과 같다.

$$P(trad.) = P_{pht\_traditional\_read} \times N_{inst} + P_{pht\_traditional\_write} \times N_{branch} + P_{btb\_read} \times N_{inst} + P_{btb\_write} \times N_{btb\_mispredictions}$$

기존 분기 예측기의 경우 PHT와 BTB는 명령어가 인출될 때마다 읽기 접근이 이루어지므로 읽기 접근에 소모되는 전력의 합은 한 번 읽는데 소모되는 전력과 전체 명령어의 수를 곱함으로써 얻을 수 있다. 인출되었던 명령어가 분기 명령어로 판정되면 PHT의 해당 엔트리 정보를 변경하므로,

PHT의 쓰기 접근에 소모되는 전력의 합은 한 번 쓰는데 소모되는 전력과 분기 명령어 수의 곱으로 계산된다. BTB의 쓰기 접근의 횟수는 BTB를 통한 목적 주소 예측이 실패한 경우에 발생하므로 BTB의 쓰기 접근에 소모되는 전력의 합은 한 번 쓰는데 소모되는 전력과 BTB 예측 실패 횟수의 곱으로 계산된다.

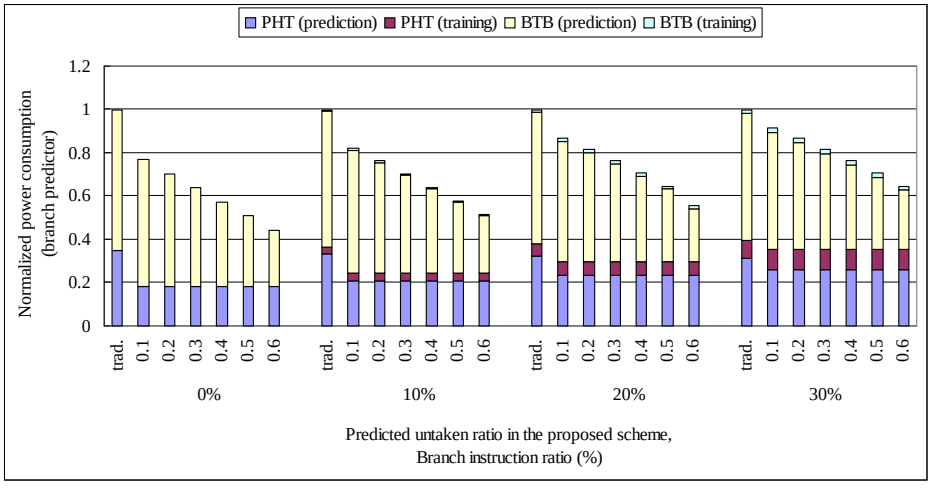
제안하는 분기 예측기의 총 전력 소모량은 다음과 같다.

$$P(prop.) = P_{pht\_proposed\_read} \times ((N_{inst} \div 2) + N_{branch}) + P_{pht\_proposed\_write} \times N_{branch} + P_{btb\_read} \times ((N_{inst} - N_{untaken\_predictions}) + N_{untaken\_previous\_btb\_hit}) + P_{btb\_write} \times N_{btb\_mispredictions}$$

제안하는 분기 예측기에서, PHT에 대한 읽기 접근 횟수는 기존 분기 예측기의 읽기 접근 횟수 (N_{inst})에 비해 줄어든 $((N_{inst} \div 2) + N_{branch})$이다. BTB에 대한 읽기 접근 횟수는 $((N_{inst} - N_{untaken_predictions}) + N_{untaken_previous_btb_hit})$로 줄어든다. 이유는 앞서 설명한 바와 같이 제안하는 분기 예측기에서는 직전 명령어가 테이큰 분기 명령어로 예측되지 않는 경우에는 PHT로부터의 예측 결과가 언테이큰인 경우에는 BTB를 접근하지 않게 되고, 직전 명령어가 테이큰 분기로 예측되는 경우에는 PHT의 예측 결과에 상관 없이 항상 BTB 접근을 하게 되기 때문이다. PHT와 BTB에 대한 쓰기 접근 횟수는 기존 분기 예측기와 제안하는 분기 예측기에서 동일하다.

(그림 5)는 분석 모델을 이용하여 기존 분기 예측기와 제안하는 분기 예측기의 전력 소모량을 비교한 결과이다. 그래프에서 보는 바와 같이 전체 명령어에 대한 분기 명령어의 비율과 PHT의 언테이큰 예측 비율을 달리하면서 제안하는 분기 예측기의 전력 소모량을 계산하였다. 그래프의 Y축은 같은 비율의 분기 명령어를 가정한 기존 분기 예측기의 전력 소모량에 정규화한 전력 소모량을 보여준다. 기존 분기 예측기가 소모하는 전력은 PHT가 언테이큰으로 예측하는 명령어의 비율에 관계 없이 항상 일정한 값을 유지하기 때문에 제안하는 분기 예측기에 대해서만 PHT의 언테이큰 예측 비율을 달리하면서 결과를 보인다. 그래프에서 *PHT(prediction)*과 *PHT(training)*은 각각 PHT에 대한 읽기에 소모된 전력의 합과 쓰기에 소모된 전력의 합을 나타내고, *BTB(prediction)*과 *BTB(training)*은 각각 BTB에 대한 읽기 전력 소모의 합과 쓰기 전력 소모의 합을 나타낸다. 일반적으로 대부분의 응용 프로그램 에서 분기 명령어의 비율은 0 ~ 30% 이고, PHT의 언테이큰 예측 비율은 0.1~0.6 (10 ~ 60%) 사이의 값을 가진다[16][17]. 이에 따라, 이러한 범위 내에서 각 비율을 달리하면서 결과를 비교하였다. 즉, 그래프의 X 축에서 0%, 10%, 20%, 30%는 응용 프로그램에서 분기 명령어의 비율을 나타내고, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6은 PHT의 언테이큰 예측 비율을 나타낸다. 그래프에서 trad.는 제안하는 저전력 기법을 적용하지 않은 기존의 분기 예측기를 의미한다. 대부분의 응용 프로그램에서 BTB의 예측 실패 비율은 0~20% 범위에 속한다. BTB 예측 실패로 인한 BTB 쓰기 접근 횟수는 예측 실패 비율에 관계 없이 기존 분기 예측기와 제안하는 분기 예측기가 동일한 횟수를 가지므로 그래프에서 BTB의 예측 실패 비율은 10%로 고정시키고 결과를 비교하였다.

제안하는 분기 예측기에서, 분기 명령어 비율의 감소는 PHT의 읽기 접근 횟수 감소로 이어지고, PHT의 언테이큰 예측 비율 증가는 BTB에 대한 읽기 접근 횟수를 감소시키는 결과를 야기한다. 그러므로, 분기 명령어의 비율이 감소하고 PHT의 언테이큰 예측 비율이 높아질수록 제안하는 분기 예측기는 기존의 분기 예측기와 비교하여 보다 높은 전력 효율성을 보인다. BTB 예측 실패로 인한 BTB 쓰기 접근 횟수는 기존 분기 예측기와 제안하는 분기 예측기에서 동일하므로 BTB 쓰기 접근에 소모되는 전력은 두 구조에서 동일하다. PHT의 경우에는 표 1에서 보인 바와 같이 한 번 쓰는데 소모되는 전력이 제안하는 PHT가 기존 PHT에 비교하여 조금 크기 때문에, PHT 쓰기 접근에 소모되는 전력은 제안하는 분기 예측기에서 약간 많다는 것을 확인할 수 있다.



(그림 5) 분석 모델을 이용한 분기 예측기의 전력 소모량 비교

(그림 5)에서 보이는 분석 결과에 따르면, 기존의 분기 예측기의 경우 PHT 읽기 접근에 소모되는 전력은 총 전력 소모의 31~35%를 차지하고, BTB 읽기 접근에 소모되는 전력은 총 전력의 58~65%를 차지한다. 제안하는 분기 예측기는 이와 같이 큰 비중을 차지하는 읽기 접근 (예측을 위한 접근)에 사용되는 전력을 줄이기 위해 제안하는 구조이다. 제안하는 분기 예측기의 PHT 읽기 접근에 소모되는 전력은 분기 명령어의 비율에 따라 기존 분기 예측기와 비교하여 17~48% 감소한다. 또한, 제안하는 분기 예측기는 BTB 읽기 접근에 소모하는 전력은 PHT의 언테이큰 예측 비율에 따라 7~60% 감소시킨다. 분기 예측기에서의 전체 전력 소모를 비교하면 제안하는 구조는 기존의 구조에 비해 9~56%의 소모 전력을 감소시킨다.

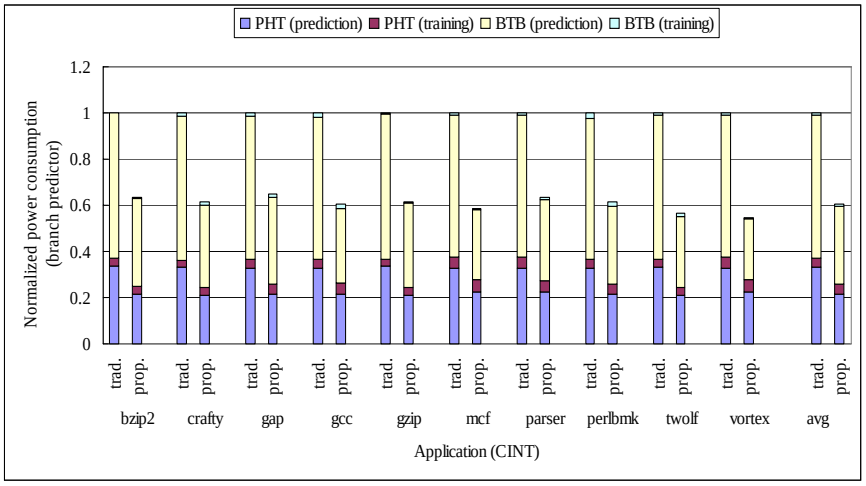
5.2 모의 실험 결과

SimpleScalar를 이용하여 여러 종류의 응용 프로그램에 대하여 모의 실험을 수행하여 다음과 같은 결과를 얻었다. (그림 6)과 (그림 7)은 각각 정수 응용 프로그램과 실수 응용 프로그램을 사용한 모의 실험 결과를 보여주고 있다. 그래프에서 Y축은 기존 분기 예측기가 소모하는 전력에 정규화한 분기 예측기의 총 소모 전력을 보여준다. 그래프의 X축에서 trad.는 제안하는 기법을 적용하지 않은 기존의 분기 예측기를 나타내고, prop.는 본 논문에서 제안된 저전력 기법을 적용한 분기 예측기를 의미한다.

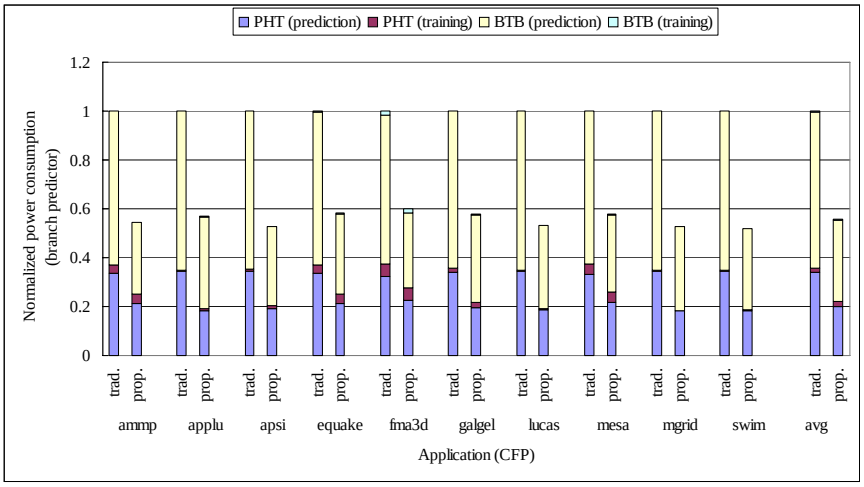
〈표 3〉 응용 프로그램 종류에 따른 분기 명령어 비율, 언테이큰 예측 비율

종 류	분기 명령어 비율	PHT 언테이큰 예측 비율
정수 응용 프로그램 (평균)	13.2 %	50.2 %
실수 응용 프로그램 (평균)	6.6 %	50.0 %

정수 응용 프로그램에 대해서 제안하는 분기 예측기는 기존 분기 예측기에 비해 평균 40%의 전력 소모를 감소시키는 결과를 보인다. PHT 예측에 필요한 전력 소모는 평균 34% 감소시키고, BTB 예측에 소모되는 전력은 평균 46%



(그림 6) 정수 응용 프로그램에 대한 분기 예측기의 전력 소모량 비교



(그림 7) 실수 응용 프로그램에 대한 분기 예측기의 전력 소모량 비교

감소시키는 결과를 보인다. 실수 응용 프로그램에 대해서 제안하는 분기 예측기는 기존 분기 예측기와 비교하여 PHT 예측에 소모되는 전력을 평균 41% 감소시키고, BTB 예측에 필요한 전력은 평균 44% 감소시킨다. 제안하는 분기 예측기는 정수 응용 프로그램에 비해 실수 응용 프로그램에서 더 많은 비율의 전력 소모를 감소시킨다. 이러한 이유는 표 3에서 보는 바와 같이 실수 응용 프로그램에서의 분기 명령어의 비율이 정수 응용 프로그램보다 낮으므로, PHT 예측에 소모하는 전력을 보다 많이 감소시킬 수 있기 때문인 것으로 생각된다. PHT의 언테이큰 예측 비율은 두 종류의 응용 프로그램에서 거의 비슷한 값을 보이므로, BTB 예측에 소모되는 전력의 감소 비율은 두 응용 프로그램에서 거의 동일하다.

모의 실험 결과, 기존 분기 예측기와 제안하는 분기 예측기는 분기 예측의 정확성과 실행 시간 측면에서 동일한 성능을 보인다. 이는 4.1절에서 기술한 바와 같이 PHT의 예측 결과는 기존 분기 예측기와 제안하는 분기 예측기에서 항상 같은 결과를 보이고, 제안하는 구조로 인해 PHT 접근 시간이 길어지는 등의 성능에 영향을 주는 현상은 나타나지 않기 때문이다.

6. 결 론

본 논문에서는, 분기 예측기를 구성하는 PHT와 BTB에 대한 읽기 접근 횟수를 줄임으로써 내장형 프로세서에 적합한 저전력 분기 예측기를 제안하였다. 제안하는 분기 예측기는, 한 번의 PHT 접근을 통해 두 개의 예측 결과를 가져오고, 직전 명령어가 테이큰 분기로 예측되지 않는 경우 BTB는 PHT의 예측 결과가 테이큰인 경우에만 접근된다. 이를 통해, 제안하는 분기 예측기는 정확도의 감소나 성능 저하 등의 문제점을 발생시키지 않으면서 소모 전력을 줄이는 장점을 보인다. 모의 실험 결과, 제안하는 분기 예측기는 기존 분기 예측기와 비교하여 35~48%의 전력을 감소시키는 결과를 보인다. 그러므로, 제안하는 분기 예측기는 향후 내장형 프로세서의 분기 예측기를 설계하는데 있어서 좋은 모델 중 하나가 될 것으로 기대된다. 본 논문에서 기술한 연구 내용을 바탕으로 향후에는 하드웨어 복잡도와 같은 구현 상의 문제점들을 정확하게 파악할 수 있는 다양한 실험을 통해 제안하는 구조를 실제 시스템에 실적용 하기 위한 연구를 진행할 계획이다.

참 고 문 헌

[1] ARM Corp., ARM1136J(F)-S, http://www.arm.com/products/CPUs/ARM1136JF-S.html.

[2] ARM Corp., ARM1156T2(F)-S, http://www.arm.com/products/CPUs/ARM1156T2-S.html.

[3] D. Parikh, K. Skadron, Y. Zhang, M. Barcella, and M. Stan, "Power Issues Related to Branch Prediction", Proc. International Conference on High-Performance Computer Architecture, pp. 233-242, 2002.

[4] S. Manne, A. Klauser, and D. Grunwald, "Pipeline Gating: Speculation Control for Energy Reduction", Proc. International Symposium on Computer Architecture, pp. 132-141, 1998.

[5] Z. Hu, P. Juang, K. Skadron, D. Clark, and M. Martonosi, "Applying Decay Strategies to Branch Predictors for Leakage Energy Savings", Proc. International Conference on Computer Design, pp. 442-445, 2002.

[6] P. Petrov and A. Orailoglu, "Low-Power Branch Target Buffer for Application-Specific Embedded Processors", Proc. Euromicro Symposium on Digital System Design, pp. 158-165, 2003.

[7] G. Palermo, M. Sami, C. Silvano, V. Zaccaria, and R. Zafalon, "Branch Prediction Techniques for Low-Power VLIW Processors", Proc. The 13th ACM Great Lakes Symposium on VLSI, pp. 225-228, 2003.

[8] M. Monchiero, G. Palermo, M. Sami, C. Silvano, V. Zaccaria, and R. Zafalon, "Power-Aware Branch Prediction Techniques: A Compiler-Hints Based Approach for VLIW Processors", Proc. The 14th Great Lakes Symposium on VLSI, pp. 440-443, 2004.

[9] D. Chaver, L. Pinuel, M. Prieto, F. Tirado, and M. C. Huang, "Branch Prediction On Demand: an Energy-Efficient Solution", Proc. International Symposium on Low Power Electronics and Design, pp. 390-395, 2003.

[10] M. C. Huang, D. Chaver, L. Pinuel, M. Prieto, and F. Tirado, "Customizing the Branch Predictor to Reduce Complexity and Energy Consumption", IEEE Micro, 23(5), pp. 12-25, 2003.

[11] D. Parikh, K. Skadron, Y. Zhang, and M. Stan, "Power-Aware Branch Prediction: Characterization and Design", IEEE Trans. Computers, 53(2), pp. 168-186, 2004.

[12] D. A. Jimenez, "Reconsidering Complex Branch Predictors", Proc. International Conference on High-Performance Computer Architecture, pp. 43-52, 2003.

[13] D. A. Jimenez, S. W. Keckler, and C. Lin, "The Impact of Delay on the Design of Branch Predictors", Proc. International Symposium on Microarchitecture, pp. 67-76, 2000.

[14] S. McFarling, Combining Branch Predictors, WRL Technical Note TN-36, Digital, 1993.

[15] Samsung Electronics, Samsung Memory Compiler, 2002.

[16] Standard Performance Evaluation Corp., SPEC CPU2000 Benchmarks, available at http://www.specbench.org/osg/cpu2000.

[17] C. Lee, M. Potkonjak, and W. Mangione-Smith,

"MediaBench: A Tool for Evaluating Synthesizing Multimedia and Communication Systems", Proc. International Symposium on Microarchitecture, pp. 330-335, 1997.

[18] D. Burger, T. M. Austin, and S. Bennett, "Evaluating future microprocessors: the SimpleScalar toolset", Tech. Report TR-1308, Univ. of Wisconsin-Madison Computer Science Dept., 1997.

김 철 홍

e-mail : cheolhong@gmail.com

1998년 서울대학교 컴퓨터공학과(학사)

2000년 서울대학교 대학원 컴퓨터공학부 (공학석사)

2006년 서울대학교 대학원 전기컴퓨터 공학부(공학박사)

2005년~2007년 삼성전자 반도체총괄 SYS.LSI사업부 책임연구원

2007년~현재 전남대학교 전자컴퓨터공학부 전임강사

관심분야 : 임베디드시스템, 컴퓨터구조, SoC 설계

송 성 근

e-mail : ssgun0@gmail.com

2004년 호남대학교 정보통신공학과(학사)

2006년 호남대학교 정보통신공학과 (공학석사)

2007년 전남대학교 전자컴퓨터공학부 박사과정 입학

관심분야 : NoC, SoC, 컴퓨터구조, 네트워크