슈퍼스칼라 프로세서에서 값 예측을 이용한 모험적 실행의 전력소모 측정 및 분석

(Measurement and Analysis of Power Dissipation of Value Speculation in Superscalar Processors)

이 상 정 † 이 명 근 †† 신 화 정 †††
(Sang-Jeong Lee) (Myoung-Kun Lee) (Hwa-Jung Shin)

요 약 최근의 고성능 슈퍼스칼라 프로세서에서는 명령어 수준 병렬성(Instruction-Level Parallelism, ILP)의 장애가 되는 명령어 간의 데이타 종속관계를 극복하기 위해 명령의 결과 값을 미리 예측하여 종속 명령들을 모험적으로 실행한다. 이러한 값 예측을 사용한 모험적 실행으로 성능은 향상되나 값 예측 테이블의 빈번한 참조와 갱신으로 부가적인 전력 소모를 요구한다.

본 논문에서는 값 예측으로 인한 성능향상과 부가적인 전력소모 간의 관계를 측정 분석한다. 또한 확신 카운터(confidence counter)를 사용한 값 예측 시도의 조정으로 모험적 실행의 정도를 조절하고, 예측 성공률이 높은 유용한 명령들만을 선택적으로 예측하여 성능을 유지하면서 부가 전력소모를 줄인다. 제안된 방식의 검증을 위해 사이클 수준 시뮬레이터에 전력소모 모델을 결합하여 프로세서의 기능수준 동작뿐만 아니라 프로세서의 전체 전력소모 및 사이클 당 전력소모도 측정할 수 있는 도구를 개발하여 검증한다.

키워드 : 모험적 실행, 값 예측, 프로세서전력소모, 슈퍼스칼라 프로세서

Abstract In recent high-performance superscalar processors, the result value of an instruction is predicted to improve instruction-level parallelism by breaking data dependencies. Using those predicted values, instructions are speculatively executed and substantial performance can be gained. It, however, requires additional power consumption due to the frequent access and update of the value prediction table.

In this paper, first, the trade-off between the performance improvement and the increased power consumption for value prediction is measured and analyzed. And, in order to reduce additional power consumption without performance loss, the technique of controlling speculative execution with confidence counter and predicting useful instructions is developed. Also, in order to prove the validity, a tool is developed that can simulate processor behavior at cycle-level and measure total energy consumption and power consumption per cycle.

Key words : Speculative execution, value prediction, processor power dissipation, superscalar processor

1. 서 론

최근의 고성능 슈퍼스칼라 프로세서는 실행될 결과를 미리 예측하여 모험적으로 명령들을 실행(speculative execution)함으로써 성능을 향상시키고 있다. 대표적인 모험적 실행으로는 조건 분기의 결과를 예측하여 제어 종속관계가 있는 명령들을 미리 인출하고 실행함으로써 지속적으로 명령을 공급하여 명령어 수준 병렬성(Instruction Level Parallelism, ILP)을 높이는 분기 예측 방식이 있다[1-3]. 또한 최근에는 명령들 간의 데이타 종속관계로 인한 성능 저하를 극복하기 위해 실행되는 명령의 결과 값을 미리 예측하고, 이후 데이타 종속관계가 있는 명령들에게 조기에 공급하여 실행시킴으로써 성능향상을 기대하는 값 예측(value prediction)방식에 관하여 연구가 진행되고 있다[4-11]. 대부분의 고성능 마이크로프로세서의 설계에 관한 연구는 명령어 수준 병렬성(Instruction Level Parallelism, ILP)을 극대화시

· 본 연구는 한국과학재단 목적기초연구(R05-2000-000-00283-0) 지원으로 수행되었음
† 정 회 원 : 순천향대학교 정보기술공학부 교수
sjlee@sch.ac.kr
†† 비 회 원 : ㈜다이알로직코리아 기술연구소
lmk77@dkpower.com
††† 종신회원 : 한양여자대학 전산정보계열 교수
hjshin@hywoman.ac.kr
논문접수 : 2003년 5월 23일
심사완료 : 2003년 8월 25일

켜서 오직 성능의 극대화를 추구하였다.

그러나 최근의 고성능 마이크로프로세서에서는 전력소모(power dissipation)가 점점 중요한 설계 제약이 되고 있다. 프로세서의 클럭 속도가 빨라지고 칩이 점점 고집적화 됨에 따라 전체 마이크로프로세서 칩의 전력소모도 이에 비례하여 증가하고 있는 실정이다. 현재 사용되고 있는 최근의 고성능 마이크로프로세서는 정상 동작 하에서 30 와트 이상의 전력을 소모하고 있는 실정이다. 따라서 프로세서 설계에서 전력소모를 줄이는 것이 점점 더 중요한 이슈가 되고 있다. 저 전력소모 프로세서의 설계는 칩 설계에서 주요 설계 이슈로 대두되었고 초기에는 회로 설계자가 주로 디바이스 수준에서 저 전력 회로의 설계나 전압 스케일링(voltage scaling) 등과 같은 다소 직접적인 방식으로 접근하였다[12-14]. 그러나 최근에 고집적, 고성능 마이크로프로세서의 대두로 기존의 방식만 가지고 전력소모를 최소화하는 것이 불충분하여 아키텍처 수준에서 전력을 줄이고자 하는 연구가 시작되고 있는 추세이다. 즉, 아키텍처 수준에서 성능향상 뿐만 적은 전력소모도 고려한 프로세서 설계에 관한 연구에 집중하고 있는 실정이다[13-16].

본 논문에서는 먼저 값 예측으로 인한 성능향상과 부가적인 전력소모 간의 관계를 측정 분석한다. 그리고 확신 카운터를 사용한 값 예측 시도의 조정으로 모험적 실행의 정도를 조절하여 전력소모를 줄일 수 있음을 보인다. 또한 예측 성공률이 높은 유용한 명령들만을 선택적으로 예측하여 성능을 유지하면서 부가 전력소모를 줄이는 기법을 연구 제안한다. 즉, 성능을 유지하면서 부가 전력소모를 최소화하기 위해 종속관계를 극복하면서 예측 성공률이 높은 유용한 명령들만을 선택적으로 예측하는 기법을 제안한다. 제안된 방식의 검증을 위해 사이클 수준 시뮬레이터에 전력소모 모델을 결합하여 프로세서의 기능수준 동작뿐만 아니라 프로세서의 전체 전력소모 및 사이클 당 전력소모도 측정할 수 있는 도구를 개발하여 검증한다.

2. 이론적 배경

이 장에서는 먼저 값 예측 및 전력소모에 관한 관련 연구와 본 논문에서 사용된 혼합형 값 예측기에 대해 소개한다.

2.1 관련연구

값 예측기는 프로세서에 의해 반복 수행되어지는 명령들의 반복되는 과거의 결과 값의 유형에 근거하여 값을 예측한다. 값 예측기의 동작은 대개 비슷한데 명령의 값 예측은 다음에 수행될 명령의 PC를 사용하여 명령을 페치할 때 동시에 예측기의 테이블을 참조하여 값을 예측한다. 그리고 이 명령의 수행 후에 결과 값이 생성되면 이 값에 근거하여 예측 테이블을 갱신하고 이 후 예측을 준비한다.

슈퍼스칼라 프로세서를 위한 값 예측기로는 여러 가지 기법들이 제안되었고 대표적인 예측기로 최근값 예측기(last value predictor)[8], 스트라이드 예측기(stride value predictor), 구문형 값 예측기(context-based value predictor) 및 혼합형(hybrid value predictor) 등이 있다[10,11].

최근값 예측기는 가장 최근에 일어난 값을 근거로 한 명령의 결과 값을 예측하는 기법으로 고정된 상수형 값을 생성하는 명령의 결과를 예측하는데 적합하다. 스트라이드 값 예측기는 최근값 예측기를 확장하여 명령의 결과값이 상수형 값 뿐 만 아니라 고정된 값(스트라이드)으로 변하는 값을 예측하기 위해 개발되었다. 즉, 한 명령에 대해 최근 수행된 2개의 결과 값의 차인 스트라이드를 구하고 다음에 이 명령의 수행 시 스트라이드 값과 가장 최근 값을 더하여 값을 예측하는 기법이다. 구문형 값 예측기는 일정한 패턴으로 변하는 값들을 예측하기 위해 명령의 수행결과로 생성된 일련의 결과값들을 테이블에 저장하고 저장된 패턴을 기준으로 다음에 생성될 값을 예측하였다. 이들 예측기들은 특정 패턴의 명령들에 대해서는 잘 동작하지만 다른 패턴에 대해서는 성능이 저하되어 최근에는 이들 예측기들을 통합한 혼합형 값 예측기를 개발하고 있는 추세이다.

최근에는 값 예측기 자체의 예측 성능 뿐만 아니라 예측 테이블의 대역폭(bandwidth) 향상, 참조지연 및 전력소모 최소화할 수 있는 값 예측기를 연구하고 있다. Lee 등은 예측 시 테이블 참조 대역폭의 향상을 위해 값 예측기의 테이블 참조 및 갱신과 페치 시 예측 값의 참조를 분리한 분리된 값 예측기(decoupled value predictor)를 제안하였다[6,7]. 즉, 값 예측기의 통해 산출된 예측 값을 트레이스 캐시에 저장하여 값 예측기와 예측 값 참조 과정을 분리하였다. Bhargava 등은 Lee의 분리된 값 예측 방식을 적용하여 테이블 참조 지연 및 전력소모를 줄일 수 있는 LEA(latency and energy aware) 값 예측 기법을 제안하였다[4]. 즉, 중앙 집중형 값 예측기를 제거하고 트레이스 캐시에 예측 값과 스트라이드 등을 저장하였다. 페치 시 트레이스 캐시로부터 예측 값과 스트라이드를 페치하고 이를 PTQ(prediction trace cache queue)를 통해 파이프라인 상에 전파하였다. 명령의 실행 후 실제 값이 계산되면 PTQ의 예측 값과 스트라이드를 참조하여 갱신한 후 트레이스 캐시에 저장하였다. 이 방식은 중앙 집중형 값 예측기를 제거함으로써 테이블 참조 지연 및 전력소모를 줄일 수 있다는 장점이 있으나 예측 성능이 우수한 구문형 값 예측기나 혼합형 값 예측기를 적

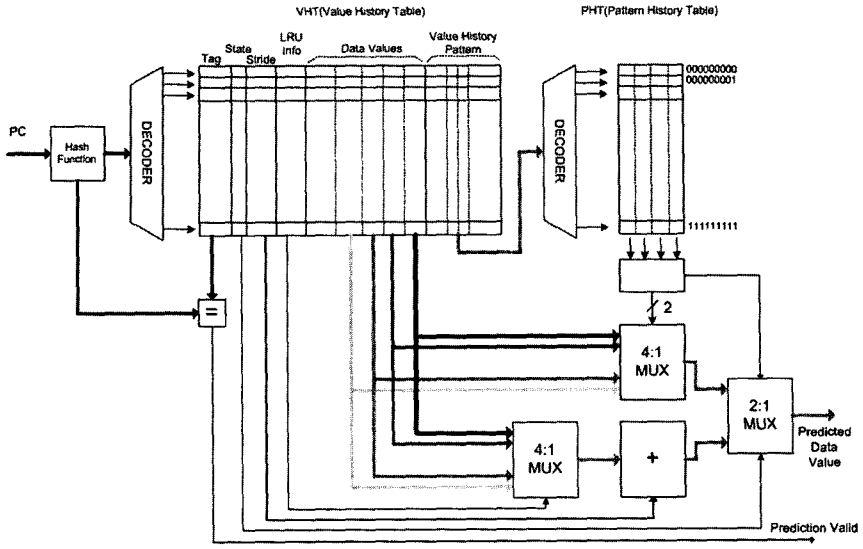


그림 1 혼합형 예측기 구성도

용할 없다는 단점이 있다. 또한 트레이스 캐시 미스 시에는 값 예측을 할 수 없어서 값 예측 성능이 저하된다는 문제점이 있다.

본 논문에서는 기존의 혼합형 예측기를 사용하여 우수한 예측 성능을 유지하면서 전력소모를 줄일 수 있는 방법을 제안한다.

2.2 혼합형 값 예측기

본 논문에서는 Wang 등이 제안한 혼합형 예측기[11]를 사용하여 값 예측기의 전력소모 최소화 방식을 연구한다. 그러나 제안된 기법은 특정 값 예측기에 관계없이 적용할 수 있다. Wang의 혼합형 예측기는 스트라이드 예측기와 구문형 값 예측기인 2-단계 값 예측기(two-level value predictor)를 혼합한 예측기이다(그림 1). 예측기는 크게 VHT(value history table)와 PHT(pattern history table)로 구성된다. VHT에는 태그 필드(Tag), 스트라이드 예측을 위한 상태 필드(State) 및 스트라이드 필드(Stride)와 구문형 값 예측을 위한 LRU 필드(LRU Info), 4개의 데이타 값 필드(Data Values)와 PHT를 인덱스하는 VHP(value history pattern)으로 구분된다. LRU 필드는 결과 값 필드에 저장된 값들의 사용된 순서에 대한 정보를 가지고 있다. VHP는 4개의 데이타 값 필드의 위치를 가리키는 2비트 인덱스로 구성되며 발생된 순서대로 데이타 값의 위치가 저장된다. PHT에는 VHT의 결과 값 필드 중 하나를 선택하기 위한 4개의 확신 카운터로 구성된다. 4개의 카운터는 4개의 데이타 값 필드에 순서대로 각각 대응된다.

먼저 예측 값을 위한 예측기의 테이블 참조(lookup) 동작은 다음과 같다. 명령어의 주소(PC)로 VHT를 인덱스하고, 해당 엔트리에 있는 결과 값의 수행 패턴을 나타내는 VHP로 PHT를 인덱스하고 PHT 엔트리의 확신 카운터에 의해 4개의 값 중 하나를 결과 값으로 예상한다. 4개의 카운터 중 가장 큰 값이 정해진 임계값(threshold value) 보다 크거나 같은 경우에 이 카운터 위치에 해당하는 데이타 값 필드의 내용을 예측값으로 사용한다. 만약 임계값 보다 크거나 같은 값의 카운터가 없다면 스트라이트 값 예측기를 사용하여 예측을 시도한다. 즉, VHT의 상태 필드가 안정 상태이면 가장 최근의 데이타 값 필드의 내용과 스트라이드를 더한 값으로 예측한다.

명령이 실행되어 결과 값이 생성되면 값 예측기의 테이블은 갱신(update)된다. 실행된 결과 값에 해당되는 VHT의 데이타 값 필드 위치에 해당하는 확신 카운터를 증가시키고 나머지 확신 카운터는 감소시킨다. 만약 실행된 결과 값이 VHT의 데이타 값 필드에 없으면 가장 오래된 데이타 값 필드 위치에 새로운 결과 값을 저장한다. 새로운 결과 값의 데이타 값 필드가 정해지면 해당 위치를 나타내는 2 비트 값이 새로운 값으로 VHP에 왼쪽으로 시프트 된다. 또한 스트라이드 예측기와 같이 VHT의 상태 필드가 갱신되고 최근에 수행된 2개의 결과 값의 차이를 구하여 스트라이드 필드에 저장한다.

3. 저전력 소모 값 예측 방식

3.1 확신 카운터 조정

값 예측을 사용한 모험적 실행으로 성능은 향상되나 값 예측 테이블의 빈번한 참조와 갱신으로 평균 9% 정도 전력소모가 증가한다(5.1 절 그림 7 참조). 본 논문

에서는 먼저 값 예측으로 인한 성능향상은 유지하면서 부가적인 전력소모를 줄이기 위해 먼저 모험적 실행의 정도를 조절하는 방법을 고찰한다. 즉, 값 예측 후 갱신되는 확신 카운터 값을 조절하여 모험적 실행의 정도를 제어한다. 예측된 명령의 수행이 완료되어 예측의 올바름을 판정한 후 이 결과에 따라 예측 테이블은 갱신된다. 대부분의 값 예측기들은 예측 정확도를 높이기 위해 확신 카운터를 가지고 있다. 올바른 예측으로 판정된 경우 카운터는 증가되고 예측이 틀린 경우 감소된다. 그리고 해당 명령의 다음 페치 시에 이 확신 카운터를 참조하여 카운터 값이 미리 정한 임계값 이상인 경우에만 예측 상태가 되어 값 예측을 시도한다. 본 논문에서 적용한 혼합형 예측기의 경우 VHT 테이블에 저장되는 4개의 데이타 값에 각각 대응되는 확신 카운터로 구성된 PHT 테이블을 갖는다. 예측된 명령의 실행 후 실제 결과 값과 같은 VHT 테이블의 데이타 값에 대응되는 확신 카운터는 증가하고 나머지 카운터는 감소된다.

이상과 같은 확신 카운터의 설계 및 적용 시에 카운트 비트 수, 증감되는 카운트 크기 및 임계값 등의 파라미터에 따라 예측 시도율이 달라지게 되어 값 예측의 성능이 달라진다. 확신 카운터의 사양을 (max, thr, inc, dec)라고 표시하자. 여기서 max는 최대 카운트 크기, thr은 임계값, inc는 증가되는 크기, dec는 감소되는 크기이다. max의 값은 카운터의 비트 크기에 따라 결정되는 값이다. 값 예측이 계속 올바르게 판정되어 카운터가 포화 상태에 이르게 되면 카운터의 값이 max가 된다. 따라서 max의 값이 커지게 되면 예측이 실패되어 카운터 값이 감소되는 경우에도 임계값 thr 보다 큰 값을 가질 확률이 높게 되어 예측기는 계속 예측을 시도하는 경향으로 남게 된다. 즉, 예측 상태에서 불예측 상태로의 전이가 느려지게 되어 값 예측기의 예측 시도율이 높아지게 된다. 임계값인 thr는 카운터의 예측 판정을 위한 기준값으로 이 값이 커지면 보다 신중한 예측을 하게 되어 값 예측기의 예측 시도율이 낮아지게 된다. 예측 성공 시 증가되는 inc 값이 커지면 보다 빨리 임계값에 도달하게 되어 예측 시도율이 높아지게 되고, 예측 실패 시 감소되는 dec 값을 크게 하면 보다 빠르게 불예측 상태로 전환하게 되어 예측 시도율을 떨어뜨린다. 일반적으로 값 예측기의 정확도가 높은 경우 예측 시도율을 높이게 되면 성능이 향상된다. 그러나 예측 시도율이 높은 경우 빈번한 예측 테이블 참조 및 예측 실패로 인하여 보다 많은 전력이 소모된다. 그리고 값 예측기의 정확도는 적용되는 벤치마크마다 달라서 확신 카운터의 파라미터 설정에 따른 성능에의 영향은 각 벤치마크마다 차이가 있다. 본 논문에서는 다양한 확신 카운터 파라미터와 벤치마크에 대해 성능과 전력소모의 영향에 대해 측정 분석하여 성능향상은 유지하면서 전력소모는 줄이는 확신 카운터의 파라미터 설정을 제시한다. 실험 분석된 결과는 이 후 5.1 절에서 기술한다.

3.2 선택적 값 예측

현재 대부분의 값 예측기들은 한 사이클에 페치되고 이슈되는 모든 레지스터 쓰기 명령(register-write instrutions)에 대해서 값 예측을 시도한다. 그러나 예측된 값은 이 후 해당 명령에 종속된 명령의 소스 오퍼랜드에 공급되어 조기 이슈되는 경우에만 성능향상에 도움이 된다. 예를들어 한 사이클에 동시에 페치되고 이슈되는 i1에서 i8까지 8개의 명령이 그림 2와 같은 데이타 종속관계를 갖는다고 가정하자(여기서 모든 명령이 1 클럭에 수행되는 ALU 명령이라 가정한다). 그림 2에서 i1, i2, i5 명령만이 각각 종속명령을 가지고 있다. 즉 이들 명령들의 예측 값들만이 각 종속명령에 공급되는 경우에만 종속명령들이 조기에 이슈되어 성능향상에 기여할 수 있다(i1-i8 명령들에 종속되고 다음 사이클에 이슈되는 명령들은 이미 i1-i8의 실행 결과값이 계산되어 더 이상 예측 값들이 필요하지 않게 된다). 그러므로 그림 2에서 같은 사이클에 페치되면서 종속된 명령들을 갖지 않는 i3, i4, i6, i7, i8 명령들은 값 예측을 할 필요가 없다. 즉 이들 명령 페치 시 값 예측 테이블을 참조할 필요가 없게 된다. 따라서 명령 페치 시 종속관계가 있는 명령들만을 감지하여 선택적으로 값 예측을 시도한다면 성능에는 영향을 미치지 않으면서 불필요한 값 예측기 테이블 참조를 제거할 수 있게 되어 테이블 참조에 의한 전력소모를 줄일 수 있다. 이 때 적용되는 명령은 한 사이클에 수행되는 명령들이고 load 명령과 같이 2 사이클 이상 소요되는 다중 사이클 명령들은 다음 사이클의 종속 명령들에도 예측 값이 유효하게 되므로 제외한다.

본 논문에서는 기존의 트레이스 캐시 구조를 사용하여 트레이스 필 유닛(trace fill unit)에서 명령들을 수집하여 트레이스를 구성할 때 데이타 종속관계를 체크하여 - 이미 다른 논문[17]에서 실행 시 최적화를 위해 종속관계 등을 체크하고 있다 - 표시하고 트레이스 캐시

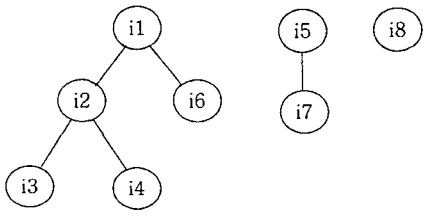


그림 2 한 사이클에 페치되는 명령들의 데이타 종속 그래프 예

에 저장한다. 트레이스 필 유닛의 동작은 프로세서 동작의 주요 실행 경로(critical path)가 아니어서 추가된 데이타 종속관계 체크는 성능이나 사이클 타임에 큰 영향을 미치지 않는다[17,18]. 이상과 같은 선택적인 값 예측을 위해 기존의 트레이스 캐시에 부가되는 하드웨어는 없고 단지 트레이스 캐시에 값 예측 시도 여부를 표시하기 위해 각 명령들의 필드에 1 비트의 유효비트가 추가된다. 따라서 트레이스 캐시가 미스되어 명령 캐시로부터 페치되는 경우에는 기존의 방식으로 모든 레지스터 쓰기 명령들에 대해 값 예측을 시도한다.

그림 3은 4개의 명령이 페치 이슈되는 머신에서 제안된 선택적 값 예측의 파이프라인 처리 구성도를 보여준다. 트레이스 캐시의 필 유닛은 명령들의 수행이 종료되고 리오더 버퍼(reorder buffer)에서 제거되는 커밋(commit) 단계에서 명령들을 수집하여 트레이스를 구성한다. 이 때 데이타 종속관계를 체크하여 트레이스 캐시의 유효비트에 저장한다. 그림 3에서는 명령 I1, I3가 유효비트가 1로 세트되어 값 예측을 시도하는 그림이다. 트레이스 캐시가 히트되어 트레이스 캐시에서 명령 페치 시 유효비트를 검사하여 값 예측기를 참조한다. 이 예에서는 I1, I3 명령의 예측값 VP1, VP3가 참조되어 종속명령들에 공급하여 이슈한다. 명령의 이슈되어 수행이 끝나 후 결과 값을 종속명령에 전파하는 라이트백(writeback) 단계에서 계산된 명령의 실제 값을 가지고 예측 테이블을 갱신한다.

제안된 선택적인 값 예측의 적용으로 평균 15% 정도의 예측 테이블 참조를 줄일 수 있었다. 선택적인 값 예측에 의한 성능 및 전력소모의 실험 및 분석 결과는 5.2절에서 기술한다.

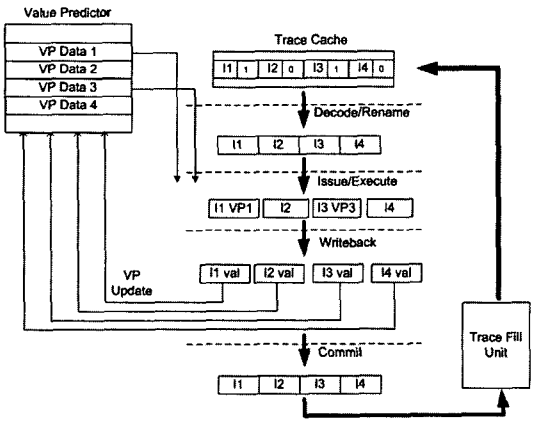


그림 3 제안된 선택적 값 예측의 파이프라인 처리 구성도

4. 전력소모 측정 도구 및 실험 환경

4.1 전력소모 측정도구

제안된 방식의 검증을 위해 사이클 수준 시뮬레이터에 전력소모 모델을 결합하여 프로세서의 기능수준 동작뿐만 아니라 프로세서의 전체 전력소모 및 사이클 당 전력소모도 측정할 수 있는 도구를 개발하였다. 그림 4와 같이 기존의 대표적인 슈퍼스칼라 프로세서의 사이클 수준 시뮬레이터인 심플스칼라 3.0(SimpleScalar 3.0)[19] 시뮬레이터에 혼합형 예측기와 Cai-Lim의 전력 소모 모델[13,15]을 결합하여 전력소모 측정 도구를 구현 개발하였다.

심플스칼라 3.0 툴 셋은 Wisconsin 대학에서 개발한 슈퍼스칼라 프로세서 시뮬레이터로 RUU(register update unit)를 명령 윈도우로 사용한다. RUU는 비순차 이슈를 지원하기 위해 디스패치된 명령을 저장하는 명령 윈도우의 기능과 명령 복구(instruction recovery)를 위해 프로그램의 순서를 유지하는 리오더 버퍼의 기능을 통합한 큐 구조의 장치이다. 전력 소모 측정을 위해 먼저 심플스칼라의 동작 부분을 다음과 같이 기능별로 세분화하였다.

* 명령 캐시, 데이타 캐시, L2 캐시, 명령 버퍼, TLB
* 명령 디코더, 디스패치 유닛(dispatch unit)
* RUU(명령 윈도우 + 리오더 버퍼)
* 레지스터 파일, 기능장치, 버스, I/O 버퍼
* 분기예측기, BTB(Branch Target Buffer), 값 예측기

세분화된 각 기능블록에 대해서 각 사이클 당 사용

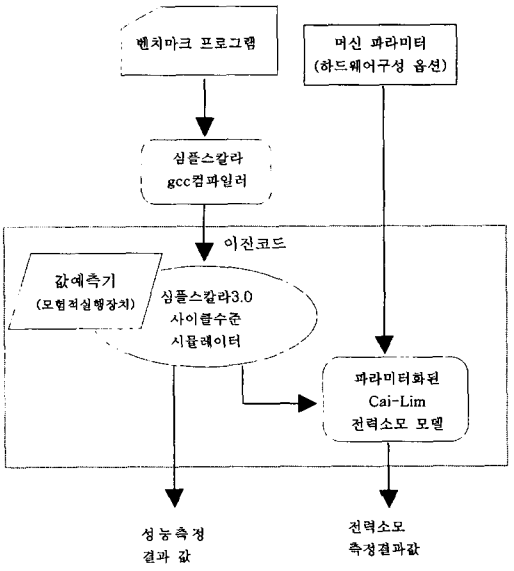


그림 4 전력소모측정 도구 구성도

빈도를 측정하는 동작 카운트 코드를 삽입하였고 이를 이용하여 프로세서의 동작 시 사이클 당 전력소모(활성 전력소모)를 측정하였다. 전력소모의 측정은 Cai-Lim의 전력모델[13]을 사용하였다. Cai-Lim 모델은 각 기능블록을 회로 종류에 따라 동적회로, 정적회로, 메모리, PLA, 클럭회로 영역으로 나누고 각 회로에 대한 단위 면적 당 전력인 전력밀도(power density)와 각 기능블록에서 회로가 차지하는 면적을 계산하여 기능블록 별 전력을 계산하였고, 이들 기능블록 별 전력을 합산하여 사이클 당 전력소모를 측정하였다. 사이클 당 소모되는 기능블록 별 전력은 다음과 같이 계산하였다.

$BAP_i = AC_i * \Sigma_j (APD_j * AR_j)$

$BIP_i = \Sigma_j (IPD_j * AR_j)$

여기서, BAP_i : 사이클 i에서 기능블록의 활성전력 (Block Active Power)

BIP_i : 사이클 i에서 기능블록의 비활성전력 (Block Inactive Power)

AC_i : 사이클 i에서 기능블록의 동작 카운트 (Activity Count)

APD_j : 회로 j에 대한 활성 전력밀도 (Active Power Density)

IPD_j : 회로 j에 대한 비활성 전력밀도 (Inactive Power Density)

AR_j : 회로 j가 기능블록에서 차지하는 면적 (Area)

전력소모는 사이클 당 소모되는 평균 전력과 전체 프로그램 수행 시 소모되는 전체 전력소모 등이 측정된다. 사이클 당 전력소모는 사이클 당 소모되는 최대 전력(peak power)을 측정하는 값으로 주로 서버용 고성능 마이크로프로세서 설계 시에 고려되는 사양이다. 즉 이 전력의 소모가 커지면 프로세서가 과열되고 냉각 시스템을 가동해야 한다. 전체 전력 소모는 배터리 등을 사용하는 모바일 기기의 임베디드 프로세서 설계 시에 고려해야 하며 배터리의 사용 시간과 관련이 있다.

본 논문에서는 개발된 전력소모 측정 도구를 사용하여 값 예측기의 부가로 인한 전력소모와, 확신 카운터와 선택적 값 예측으로 인한 성능 및 전력소모 등을 측정 분석한다.

4.2 실험 방법

표 1은 시뮬레이션 되는 프로세서의 구성에 대한 각 머신 파라미터를 보여준다. 각각 64KB의 크기를 갖는 명령 및 데이타 캐시와 분기예측기로는 8K 엔트리를 갖는 혼합형 예측기를 사용하였다(gshare와 bimodal의 혼합형 예측기)[2]. 트레이스 캐시는 4-웨이 1024 엔트리를 갖고 각 블록 당 16개의 명령을 저장한다. 혼합형 값 예측기는 8K 엔트리를 갖는 VHT 테이블에 4개의 데이타 값을 저장하여 이들 인덱스로부터 12 비트의 VHP를 생성하여 4K 엔트리의 PHT를 가리키도록 구성하였다.

표 2는 적용된 벤치마크 프로그램의 특성을 보여주는 그림이다. SPEC2000의 경우 시뮬레이션 시간을 줄이기 위해 기준 입력 세트를 수정하여 시뮬레이션 하였다[20]. 각 벤치마크 프로그램에 대해 입력값, 실행된 명령수, IPC(Instrucrions Per Clock Cycle), 분기예측 정확도 및 트레이스 캐시 미스율 등을 보여주는 표이다(값 예측을 적용않는 경우를 기준으로 측정된 값이다).

5. 성능 및 전력소모 측정 분석

5.1 확신 카운터

표 1 머신 파라미터 구성

	Parameter	Value
Processor Core	instruction window (RUU) load-store queue (LSQ) fetch/issue/commit width	256 entries 64 entries 16 instructions/cycle 16 integer units, 1-cycle latency 8 floating-point units, 2-cycle latency 8 integer multiply/divide units, 3/12-cycle latency 4 floating-point multiply/divide units, 4/12-cycle latency
Memory	120-cycle latency for the first 8 bytes, 2 cycle-latency for each 8 bytes thereafter	
Branch Predictor	branch target buffer (BTB) hybrid branch predictor return address stack (RAS)	2K entries, 2-way gshare with 8K entries,14-bit history, bimodal with 8K entries 32 entries
Caches	level-one data cache level-one instruction cache level-two unified cache	64KB, 4-way, 32 bytes/block, 2-cycle latency 64KB, 2-way, 64 bytes/bock, 2 basic blocks/access 1MB, 128 bytes/block, 12-cycle latency
Trace Cache	1024 sets, 4-way, 16 instructions/block	
Value Predictor	hybrid value predictor	8K entries VHT, 4 history data values, 4K entries PHT

표 2 벤치마크프로그램

Program	Input	Exec. Instr. (million)	IPC	Branch Accuracy	Trace Cache Miss Rate	Benchmark Set
compress	10000 e 2231	105	2.87	0.91	0.07	
gcc	jump.i	131	1.77	0.91	0.53	
go	5 9	379	1.52	0.83	0.48	SPEC95
ijpeg	tinyrose.ppm	149	3.74	0.95	0.28	integer
perl	scrabbl.pl	130	2.23	0.95	0.51	
vortex	persons.250	90	2.71	0.98	0.64	
mcf	smred.in, place	318	3.13	0.96	0.08	SPEC2000
parser	smred.in	996	2.06	0.94	0.13	integer
vpr	smred.in	72	2.00	0.85	0.46	
Average		263	2.45	0.92	0.35	

확신 카운터의 (max, thr, inc, dec) 설정에 따른 성능 및 전력소모의 영향을 측정 분석하기 위해 (3,2,1,1), (7,2,3,1), (7,6,3,1), (15,6,3,1)의 4 가지 파라미터 구성에 대해 실험하였다. 본 논문에서 적용한 혼합형 예측기는 예측 갱신 동작에서 실행된 명령의 실제 값이 VHT에 저장된 4개의 데이타 값과 불일치하는 경우에 항상 확신 카운터를 감소시키므로 dec의 값은 모두 1로 고정하였다. (3,2,1,1) 구성의 경우 2 비트 카운터로 구성되며 적은 max 값으로 인해서 예측에서 불예측의 판정 전환이 빠르게 되어 예측 시도율(prediction coverage)이 작게 된다. (7,2,3,1)과 (7,6,3,1)은 3 비트 카운터로 구성되며 (7,2,3,1) 구성이 작은 thr 값을 가져서 (7,6,3,1) 보다 빠른 예측 판정을 하게 된다. (15,6,3,1) 구성은 4 비트 카운터를 사용하며 큰 max, inc 값을 가져서 다른 구성보다도 빈번하게 값을 예측한다.

그림 5는 각 벤치마크에 대해 값 예측을 적용한 경우의 성능 향상을 보여주는 그림으로 성능이 약 3%에서 12%까지 향상되었다. 대부분의 벤치마크에서(ijpeg, mcf 제외) 각 확신 카운터에 대한 성능은 (15,6,3,1), (7,3,2,1), (7,6,3,1), (3,2,1,1) 순으로 향상되는데 이는 높은 예측 시도율을 갖는 것이 성능에 긍정적인 효과가 있음 보여준다.

그림 6은 각 확신 카운터 구성에 대한 예측 성공 및 실패율을 보여주는 그림이다. 막대의 하단이 예측 성공률이고 상단이 실패율이다. 이 두개의 합은 예측 시도율을 보여준다. 예를들어 compress (3,2,1,1)의 경우 49%의 예측 시도율을 갖는다. 즉, 전체 레지스터 쓰기 명령 중 49%가 값 예측을 시도하고 나머지 51%는 확신 카운터의 불예측 상태 판정으로 값 예측을 하지 않는다. 그리고 49% 예측 시도율 가운데 예측 성공률이 33%,

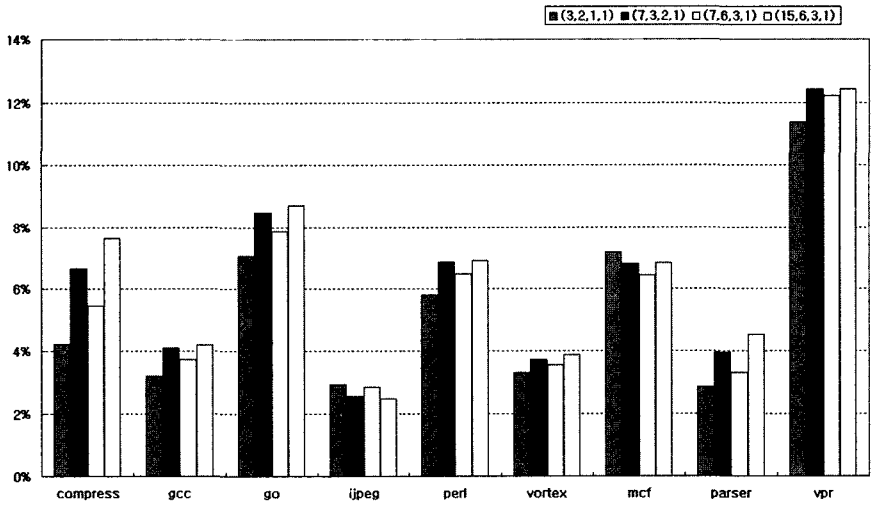
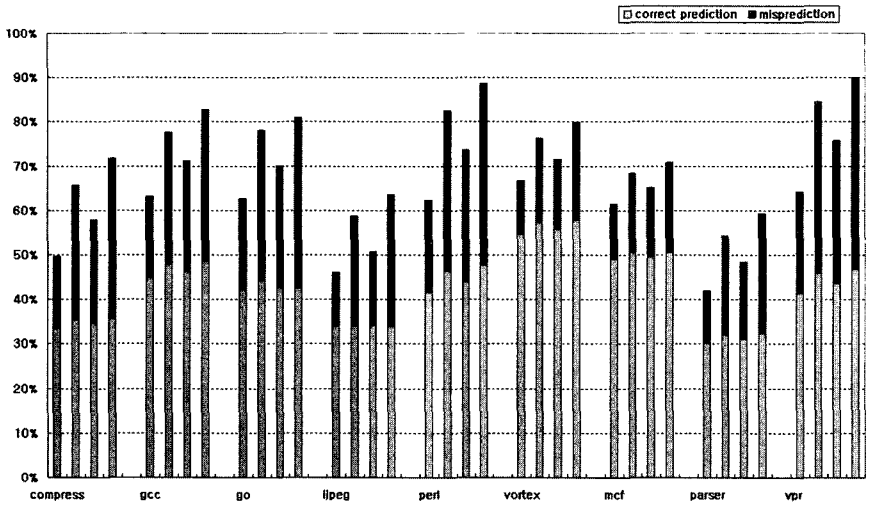


그림 5 값 예측기의 확신 카운터 구성에 따른 값 예측 성능 향상



각 벤치마크의 막대는 왼쪽부터 (3,2,1,1) (7,3,2,1) (7,6,3,1) (15,6,3,1) 순이다.

그림 6 값 예측 정확도 및 미스율

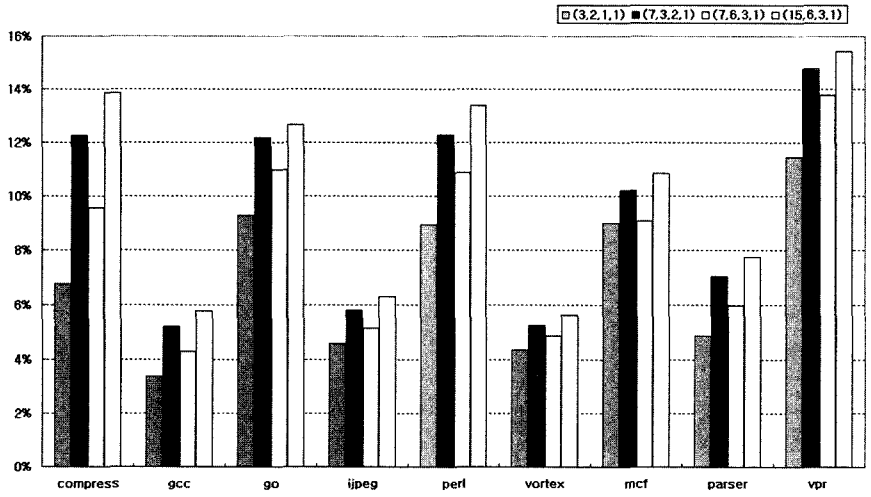


그림 7 값 예측으로 인한 사이클 당 평균 전력소모 증가율

실패율이 16%임을 보여 준다. 예측 시도율은 (15,6,3,1), (7,3,2,1), (7,6,3,1), (3,2,1,1) 구성 순으로 큰 값을 갖고 이는 그림 5의 성능 순과 거의 일치한다.

그림 7은 값 예측을 적용하였을 경우 증가되는 사이클 당 평균 전력소모를 보여 주는 그림으로 벤치마크에 따라 3%에서 15%까지(평균 9%) 사이클 당 평균 전력소모가 증가됨을 나타낸다. 이는 주로 값 예측기의 테이블 참조 및 갱신으로 인해 부가되는 전력소모이다. 사이클 당 평균 전력소모 증가는 전체적으로 예측 시도율과 일치하여 (15,6,3,1), (7,3,2,1), (7,6,3,1), (3,2,1,1) 구성 순으로 큰 전력소모를 보였다.

값 예측으로 인한 성능 향상은 유지하면 사이클 당 전력소모는 최소로 하는 확신 카운터의 구성을 찾기 위해 사이클 당 전력소모 증가율을 성능 향상율로 나누어서 사이클 당 전력소모 증가 대 성능 향상 비율을 계산하였고(그림 8), 이 값이 작을수록 성능 향상을 유지하면서 작은 전력소모의 경향을 나타낸다.

그림 8에서 전체적으로 (3,2,1,1)이 구성이 상대적으로 가장 작은 값을 보여서 고성능 프로세서에서 값 예측으로 인한 성능 향상을 추구하면서 사이클 당 전력소모 증가는 줄일 수 있음을 알 수 있다. 즉, 다시 말해서 확신 카운터 구성 시 예측 판정 상태와 불예측 판정 상태

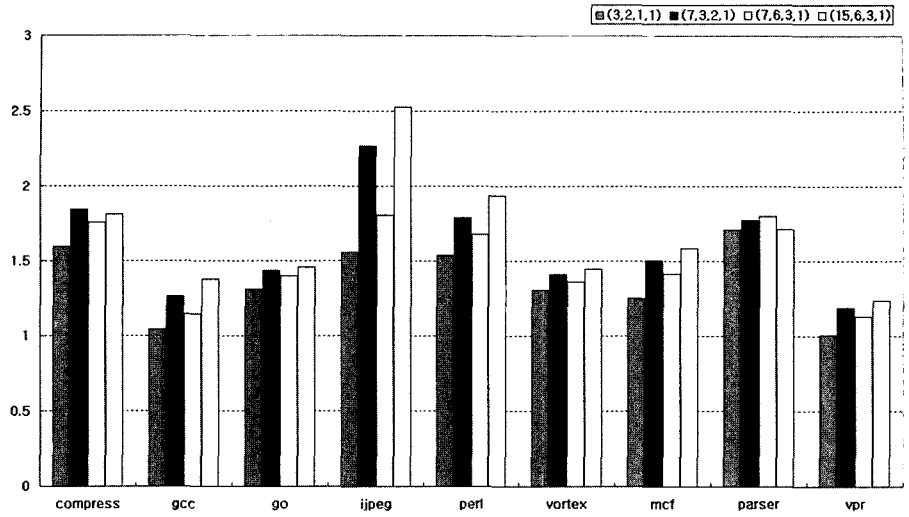


그림 8 사이클 당 전력소모 증가 대 성능 향상 비율

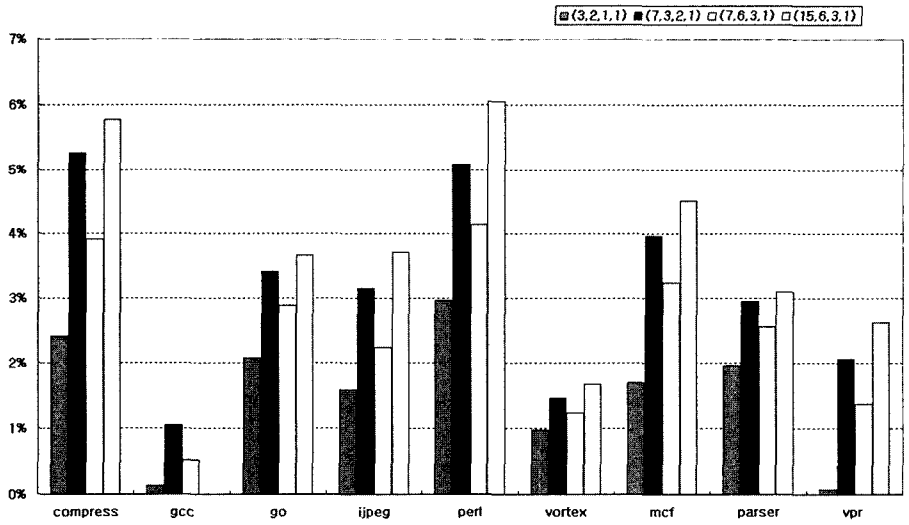


그림 9 값 예측으로 인한 전체 전력소모 증가율

의 전환을 빠르게 하는 구성이 값 예측으로 인한 전력소모를 줄이는데 중요한 요소가 된다. 또한 vpr, go, mcf 등의 벤치마크는 값 예측으로 인한 높은 성능향상을 보이면도 상대적으로 작은 전력을 소모하여 우수한 사이클 당 전력소모 증가 대 성능 향상 비율을 보였다.

앞서 언급한 바와 같이 고성능 프로세서 설계 시 우선 고려되는 전력소모는 사이클 당 전력소모이다. 그러나 최근 임베디드 시스템의 사양이 고성능화 되는 추세이어서 값 예측으로 인한 전체 전력소모 증가도 측정하였다. 그림 9는 값 예측으로 인한 전체 전력소모 증가율을 보여주는 그림이다. 전체 전력소모의 증가율이 평균 3% 정도로 사이클 당 전력소모 증가율 보다 작은 값을 갖는다. 이는 값 예측으로 인한 모험적 실행으로 명령들의 이슈가 빨라지고 따라서 분기명령의 이슈 및 실행도 전체적으로 빨라져서 분기예측이 틀린 경우에 조기에 판정되어 잘못 예측된 경로의 페치 및 수행을 조기에 차단하기 때문에 전체적으로 불필요한 명령의 페치 및 이슈가 줄었기 때문이다. 실제로 모든 벤치마크의 전체 수행된 명령이 값 예측을 적용하지 않은 경우 평균 263M(million)개의 명령을 페치하여 수행한 반면 (3,2,1,1) 구성의 확신 카운터를 가지고 값 예측을 적용한 경우 평균 253M 개의 명령만을 페치하고 수행하여

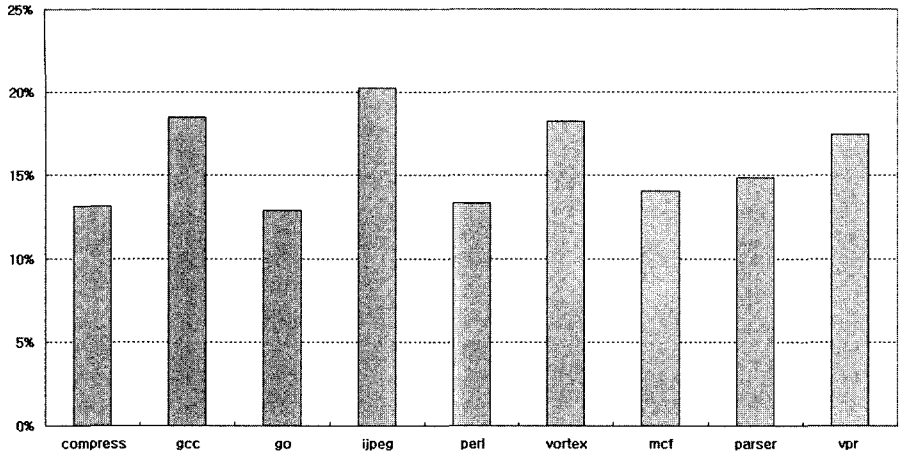


그림 10. 선택적 값 예측으로 인한 예측 테이블 참조 감소율

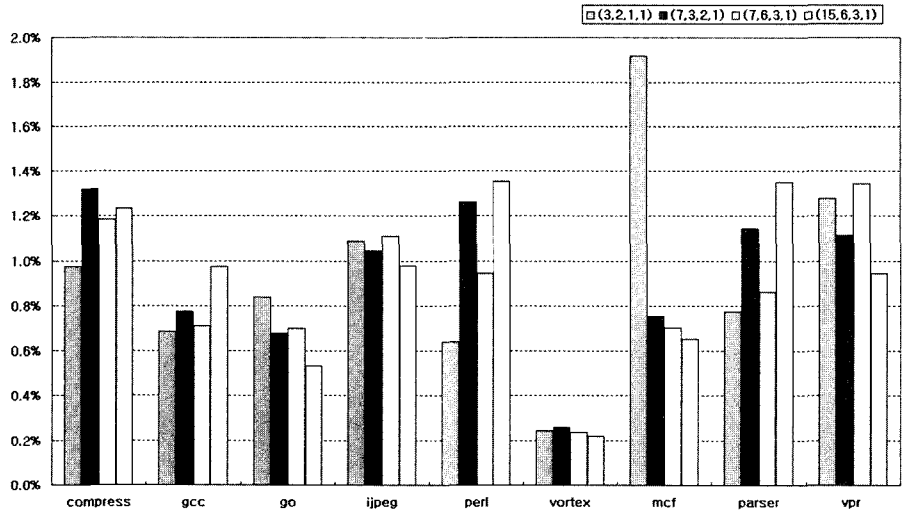


그림 11. 선택적 값 예측으로 인한 사이클 당 평균 전력소모 감소율

평균 10M 개의 명령들의 페치 및 수행이 줄었다.

5.2 선택적 값 예측

그림 10은 본 논문에서 제안한 선택적인 값 예측을 적용하였을 경우 예측 테이블 참조 감소율을 나타낸다. 즉, 트레이스 캐시에서 페치될 때 다중 사이클 명령이 아니면서 데이타 종속관계가 없는 명령이어서 값 예측을 시도하지 않는 명령들의 비율을 의미한다. 그림 10에 나타난 바와 같이 전체 레지스터 쓰기 명령 중에서 평균 15%의 명령들의 리던던트한 값 예측을 줄일 수 있었다.

그림 11은 선택적인 값 예측을 적용하지 않는 경우와 비교하여 적용하였을 경우 감소되는 사이클 당 평균 전력소모 감소율을 나타내고 평균 1%(0.2%-1.9%) 정도 전력소모가 줄어들었다. 이는 주로 값 예측 시 불필요한 예측 테이블의 참조를 제거함으로써 얻어진 전력소모 감소이다. 각 확신 카운터에 대한 감소율은 대부분 예측 시도율의 패턴을 따르지만 go, mcf 등에서는 (3,2,1,1)의 감소율이 두드러지는데 이는 빠른 상태 전환으로 상대적으로 빈번하게 예측 상태에 놓이면서 데이타 종속관계를 갖지 않는 명령들을 예측에서 제외하였기 때문이다.

그림 12는 선택적인 값 예측을 적용하지 않는 경우와 비교하여 적용하였을 경우 감소되는 전체 전력소모 감소율을 나타내고 평균 0.8%(0.01%-1.6%) 정도 전력소모가 줄어들었다.

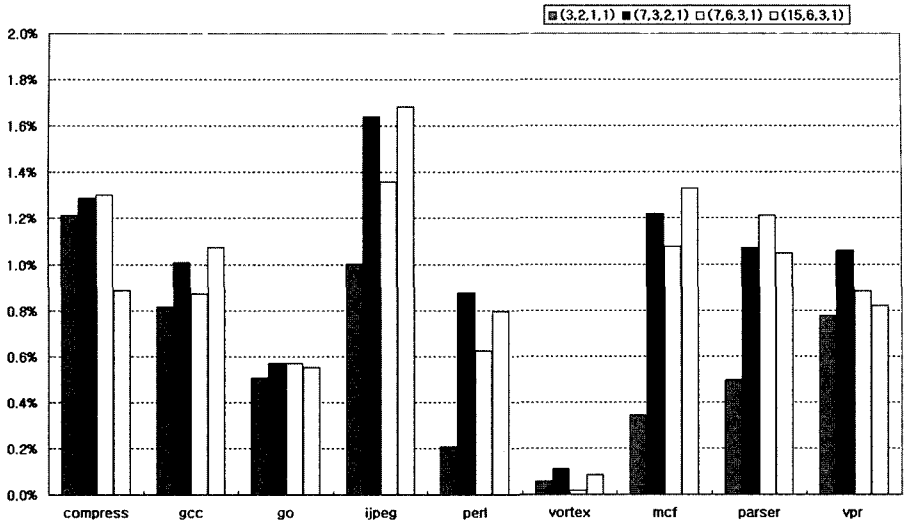


그림 12. 선택적 값 예측으로 인한 전체 전력소모 감소율

6. 결 론

본 논문에서는 사이클 수준 시뮬레이터에 전력소모 모델을 결합하여 프로세서의 기능수준 동작뿐만 아니라 프로세서의 전체 전력소모 및 사이클 당 전력소모도 측정할 수 있는 도구를 개발하여 값 예측으로 인한 성능향상과 부가적인 전력소모 간의 관계를 측정 분석하였다.

첫째, 확신 카운터의 다양한 카운터 파라미터의 조정으로 값 예측 시도율을 조절하여 성능향상과 모험적 실행의 정도를 제어함으로써 성능향상을 유지하면서 전력소모를 줄이는 확신 카운터 구성을 제시하였다. 실험 결과 각 확신 카운터에 대한 성능과 전력소모량은 (15,6,3,1), (7,3,2,1), (7,6,3,1), (3,2,1,1) 순으로 높았는데 전체적으로 예측 시도율과 일치하였다. 이 중에서 (3,2,1,1)이 구성이 예측 판정 상태와 불예측 판정 상태의 전환이 빨라서 값 예측으로 인한 성능 향상을 추구하면서 사이클 당 전력소모 증가는 가장 적었다. 둘째, 모든 레지스터 쓰기 명령에 대해 값 예측을 시도하지 않고 한 사이클에 트레이스 캐시에서 페치되는 명령들 중 데이타 종속관계가 있는(즉, 예측 값이 필요한) 명령들만을 선택적으로 값 예측을 하여 성능을 유지하면서 부가 전력소모를 줄였다. 실험 결과 전체 레지스터 쓰기 명령 중에서 평균 15%의 명령들의 리던던트한 값 예측을 줄여서 불필요한 예측 테이블 참조를 줄였고 그 결과 최대 1.9%까지 전력소모를 줄일 수 있었다.

참 고 문 헌

[1] D.Grunwald,A.Klauser,S.Manne and A. Pleszkun, "Confidence Estimation for Speculation Control," Proceedings of the 25th International Symposium on Computer Architecture (ISCA-25), June 1998.

[2] S.McFarling, "Combining Branch Predictors," Technical Report TN-36, Digital Western Research Laboratory, June 1993.

[3] T.Yeh and Y.Patt, "Two-level Adaptive Branch Prediction," Proceedings of the 24th International Symposium Microarchitecture(MICRO-24), Nov. 1991.

[4] R.Bhargava and L.John, "Latency and Energy Aware Value Prediction for High-Frequency Processors," Proceddings of 16th ACM International Conference on Supercomputing, pp. 45-56, June, 2002.

[5] B.Calder, G.Reinman and D.Tullsen, "Selective Value Prediction," Proceedings of the 26th International Symposium on Computer Architecture(ISCA-26), May 1999.

[6] Sang-Jeong Lee and Pen-Chung Yew, "On Table Bandwidth and Its Update Delay for Value Prediction on Wide-Issue ILP Processors," IEEE Transaction on Computers, Vol.50 No.8, p.847-852, Aug. 2001.

[7] Sang-Jeong Lee and Pen-Chung Yew, "On Augmenting Trace Cache for High-Bandwidth Value Prediction," IEEE Transaction on Computers, Vol.51, No.9, p.1074-1088, Sept. 2002.

[8] M.Lipasti and J.Shen, "Exceeding the Limit via Value Prediction," Proceedings of the 29th International Symposium on Microarchitecture(MICRO-29), Dec. 1996.

[9] B.Rychlik, J.Faistl, B.Krug, and J.Shen, "Efficacy

and Performance Impact of Value Prediction," Parallel Architectures and Compilation Techniques (PACT98), Paris, Oct. 1998.
[10] Y.Sazeides and J.Smith, "The Predictability of Data Values," Proceedings of the 30th International Symposium on Microarchitecture(MICRO-30), Dec. 1997.
[11] K.Wang and M.Franklin, "Highly Accurate Data Value Predictions using Hybrid Predictor," Proceedings of the 30th International Symposium on Microarchitecture(MICRO-30), Dec. 1997.
[12] M.Borah, R.Owens and M.Irwin, "Transistor sizing for low power CMOS Circuits," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 15 No. 6, 1996.
[13] G. Cai, "Architectural Level Power/Performance Optimization and Dynamic Power Estimation," Proceedings of the CoolChips Tutorial, An Industrial Perspective on Low Power Processor Design in conjunction with the 32th Annual International Symposium on Microarchitecture (MICRO-32), 1999.
[14] C.Leung, D.Brooks, M. Martonosi, and D.Clark, "Power-Aware Architecture Studies: Ongoing Work at Princeton," Proceedings of Power-Driven Microprocessor Design Workshop, June 1998.
[15] S. Ghiasi and D. Grunwald, "A Comparison of Two Architectural Power Models," Proceedings of the Workshop on Power-Aware Computer Systems in conjunction with ASPLOS-IX, Nov., 2000.
[16] S.Manne,A.Klauser and D.Grunwald, "Pipeline Gating: Speculation Control for Energy Reduction," Proceedings of the 25th International. Symposium on Computer Architecture(ISCA-25), June 1998.
[17] D.Friendly, S.Patel, and Y.Patt, "Putting the Fill Unit to Work : Dynamic Optimizations for Trace Cache Microprocessors," Proceedings of the 31st International Symposium on Microarchitecture (MICRO-31), Dec 1998.
[18] E.Rotenburg, S.Bennett, and J.Smith, "A Trace Cache Microarchitecture and Evaluation," IEEE Transaction on Computers, Vol.48 No.2, Feb. 1999.
[19] D.Burger and T.Austin, *The SimpleScalar Tool Set, Version 2.0*, Technical Report CS-TR-97-1342, University of Wisconsin, Madison, June 1997.
[20] A.KleinOsowski, J.Flynn, N.Meares, and D.Lilja, "Adapting the SPEC 2000 Benchmark Suite for Simulation-Based Computer Architecture Research," Workshop on Workload Characterization held in conjunction with International Conference on Computer Design, Sept., 2000.

이 상 정

1983년 2월 한양대학교 공과대학 전자공학과 공학사. 1985년 2월 한양대학교 대학원 전자공학과 공학석사. 1988년 8월 한양대학교 대학원 전자공학과 공학박사 1988년 9월~현재 순천향대학교 공과대학 정보기술공학부 교수. 1999년 2월~2000년 1월 미국 University of Minnesota 방문교수. 관심분야는 고성능 프로세서 설계, 컴파일러, 네트워크 응용

이 명 근

1992년 2월 순천향대학교 컴퓨터학부 공학사. 2001년 2월 순천향대학교 전산학과 공학석사. 2001년 2월~현재 순천향대학교 전산학과 박사과정. 2001년 7월~현재 ㈜다이알로직코리아 기술연구소. 관심분야는 VoIP, 네트워크 응용, 컴퓨터구조

신 화 정

1985년 2월 이화여자대학교 자연과학대학 전자계산학과 이학사. 1987년 2월 이화여자대학교 대학원 전자계산학과 이학석사. 2001년 2월 이화여자대학교 과학기술대학원 컴퓨터학과 공학박사. 1993년 3월~현재 한양여자대학 전산정보계열 교수. 관심분야는 프로그래밍언어론, 컴파일러, 컴퓨터구조