AE32000에서의 효율적인 분기 예측 기법

정주영[0] 김현규 오형철*
고려대학교 대학원 전자정보공학과
*고려대학교 자연과학대학 공학부
{juyoung, babyworm, hyeong}@atlas.korea.ac.kr

Effective Branch Prediction Schemes in AE32000

Ju-Young Jung[0] Hyun-Gyu Kim Hyeong-Cheol Oh*
Dept. of Elec. & Info. Eng., Graduate School, Korea University, Korea
*School of Engineering, Korea University, Chochiwon, Chung-Nam 339-700, Korea

요 약

본 논문에서는 AE32000 프로세서에 적용 가능한 효율적인 분기 예측 기법에 관하여 연구하였다. 실험 결과, 내장형 응용분야에서의 비용 효율성이란 측면에, AE32000 프로세서에서는 1비트의 분기 예측기와 한 개의 엔트리를 갖는 BTB(Branch Target Buffer)를 사용하는 것이 가장 적합함을 관찰하였다. 또한, 분기 목적 주소에서 나타나는 LERI 명령을 폴딩하여 분기 손실을 줄이는 방안은, BTB와 LERI 폴딩 유닛을 사용하는 설계에서, 가져오는 성능 향상이 미미함을 확인하였다.

1. 서 론

본 논문에서는, 내장형 응용을 목표로 코드 밀도를 높이기 위하여 고안된 명령어 집합 아키텍처인 EISC(Extendable Instruction Set Computer) 아키텍처[1]를 기반으로 하는 AE32000 프로세서[2]를 위한 효율적인 분기 예측 기법을 다룬다.

AE32000은 32비트 내장형 프로세서 코어로서 5단계 파이프라인 및 32비트 명령어/데이터 버스를 지니는 하바드 아키텍처를 채용하였으며, 8/16/32비트 연산을 지원하고, 32x32=64 곱셈기와 DCT연산이나 DSP 연산에서 많이 사용되는 32x32+64=64 MAC 연산기를 내장한다. 또한, 7개의 특수 레지스터와 16개의 범용 레지스터를 가지며, 최대 4개까지의 보조 프로세서를 통한 확장이 가능하고, 총 14가지의 조건분기를 지원하며, 한번에 최대 8개까지의 레지스터를 push/pop할 수 있는 push/pop Reg. List명령을 지닌다. 그림 1은 AE32000 마이크로 아키텍처의 개요인 바, EISC 구조 특유의 독립적인 명령어인 LERI(Load Immediate to Extension Register) 명령어가 가져오는 성능 손실을 피하기 위하여 명령어 폴딩 유닛을 갖는다. LERI 명령어는 16비트로 제한된 크기의 명령어에서 필드의 확장을 위하여 EISC구조에서 제공되는 명령어로서, 이 명령어의 추가로 인하여 프로세서의 성능이 저하될 수 있다[2].

기존의 내장형 마이크로 프로세서들은 하드웨어 비용상의 제약으로 분기 손실 감소 기법을 사용하지 않았지만, 내장형 응용분야에서 요구되는 성능이 현저히 향상됨에 따라, 오늘날에는 기존의 간단한 기법들뿐 아니라, 카운터를 사용하거나 분기 주소 계산을 위해 전용 명령어를 사용하는 기법 등, 내장형 응용에 적합한 저비용의 분기 손실 감소 기법들을 개발하여 사용하고 있다[3][4]. 그런데, 이들 기 개발된 분기 손실 감소 기법들은, LERI 명령어를 갖는 프로세서의 설계에 적용하기에 지나치게 큰 추가 비용을 가져오거나, EISC 구조가 갖는 장점을 감소시켜, AE32000의 설계에는 적용하기 어렵다.

본 논문에서는 AE32000에 적합한 분기 예측 기법을 알아보기 위하여 비교적 간단한 기존의 분기 손실 감소 기법들을 적용하여 실험한 후, 분기 예측 방안에 따른 성능 변화와 비용 효율성을 비교 분석한다. 또한, 명령어 폴딩 유닛을 포함하는 효율적인 분기 처리기의 구현 방안과 분기 시에 나타나는 LERI 명령어를 효율적으로 처리하는 방안들을 모색한다.

본 논문의 구성은 다음과 같다. 제2절에서는 본 논문에서 실험을 수행하고 평가하는 방법을 설명하고, 제3절에서는 수행한 분기 예측 기법에 따른 실험 결과를 보여주며, 제4절에서 결론을 맺도록 한다.

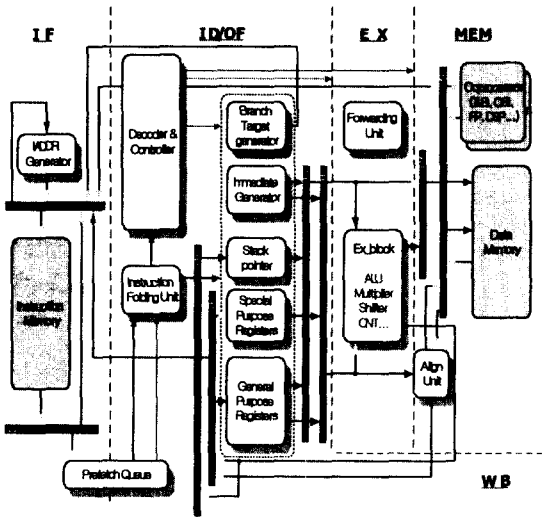

그림 1 AE32000의 마이크로 아키텍처 블록 다이어그램

2. 실험 방법 및 평가 방법

본 논문에서는 AE32000의 마이크로 아키텍처 수준의 성능 측정 및 기능 검증 시 기준 비교 모델로 사용하기 위해 제작된 주기 정밀형(cycle-accurate) 시뮬레이터인 ESCASim[5]을 사용하였다. 각 분기 예측 모델을 지원하기 위하여 BTB(Branch Target Buffer)[6]를 class형태로 만들어 기본적인 분기 관련 함수를 작성하였으며, 명령어 디코딩(ID)단계와 명령어 패치(IF)단계의 함수를 재정의(overloading)하여 실험을 수행하였다. 성능 측정에는 Dhrystone 2.1 벤치마크 프로그램이 사용되었으며, 머신 초기화 코드 부분은 제외한 후 성능 측정을 수행하였다. 성능 비교를 위한 기본 머신으로써 LERI 명령

어 폴딩 및 분기 예측이 모두 포함되지 않은 머신을 사용하였으며, 하드웨어 비용 비교에 있어서는 LERI 명령어 폴딩 유닛이 포함된 모델을 기준으로 하여 비교를 수행하였다. 내장 목적의 AE32000에 대한 각 분기 예측 기법의 적합성을 비교하기 위해서 각각의 적용된 기법으로 인한 성능 향상 및 추가되는 하드웨어 비용을 요소로 하는 비용 효율성이란 척도를 다음과 같이 정의하여 사용하였다.

Cost Effectiveness(%)
= *Performance Improvement*(%) ÷ *H/W Cost*(%)

즉, 비용 효율성은 하드웨어 추가분에 대한 성능 증가분의 비율로서, 본 논문에서는 비용 효율성이 1보다 작은 설계는 하드웨어 비용이 특히 중요한 내장형 응용분야에서 바람직하지 않은 것으로 평가하였다.

추가되는 하드웨어 비용의 계산은 3개까지의 LERI 명령어를 폴딩하는 방안과 8엔트리(entry)의 명령어 선인출 큐(Prefetch Queue)를 포함하는 RTL 수준의 AE32000 모델을 Synopsys로 합성한 후, 이를 기준으로 그 비율을 평가하였다. 일부 합성하지 못한 부분에 대해서는 Cell Library의 Gate Count를 참조하여 앞에서 합성된 결과로부터 가감하여, 약 40,000 게이트의 기본 하드웨어 비용을 구하였다.

3. 성능 평가

3. 1 LERI 명령어 폴딩 기법

AE32000에서는 명령어 선인출 기법과 크로스바 스위치를 이용한 LERI 명령어 폴딩 방안을 사용한다.[2] 그림 2에서는 기본 머신에 대한 IPC 향상율과 하드웨어 비용 증가율 및 비용 효율성을 2가지 LERI 명령어 폴딩 방안에 대하여 보였다. 그림 2에서 선인출 큐의 크기가 2이상인 경우 성능 향상이 없지만, 이는 이상적인 외부 메모리 시스템을 가정하였기 때문이며, 일반적인 경우에 큰 선인출 큐를 운용하는 경우 캐쉬 미스에 의한 성능 저하를 줄여주므로 성능에 더 좋은 결과를 나타낸다.

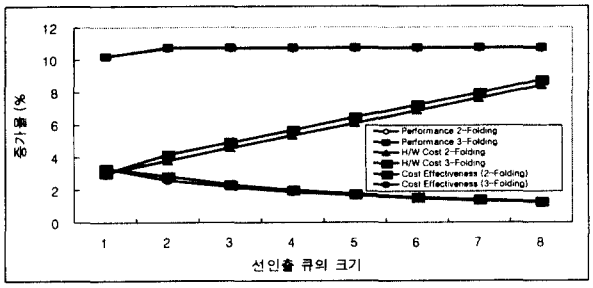

그림 2. Prefetch queue의 크기와 LERI 명령어 폴딩 방안에 따른 IPC 향상율(%), 하드웨어 비용 증가율(%), 비용 효율성(%)

3. 2 Predict-Taken 기법

일반적인 조건 분기 명령에서 taken의 비율이 60%이상을 차지하며, 내장형 응용에서는 그 비율이 더 증가하는 것으로 알려져 있다. 따라서, 가장 간단한 형태로서 BTB를 이용한 predict-taken기법을 채택하였다. 이 기법을 적용한 후 BTB의 크기를 변경시키면서 실험을 수행한 결과 그림 3과 같은 성능 향상을 얻을 수 있었다. 분기 목적 주소에 LERI가 존재하는 경우 분기 예측의 성공이 LERI의 폴딩 비율을 향상시키므로 LERI 명령어의 폴딩 비율을 함께 측정하여 나타내었다. 그림 4는 각 기법에 따른 하드웨어 소모량과 성능 증가율, 이로 인한 비용 효율성을 나타내었다. 비용 효율성이 1이하인 경우는 내장형 응용분야에서는 바람직하지 않음을 의미한다.

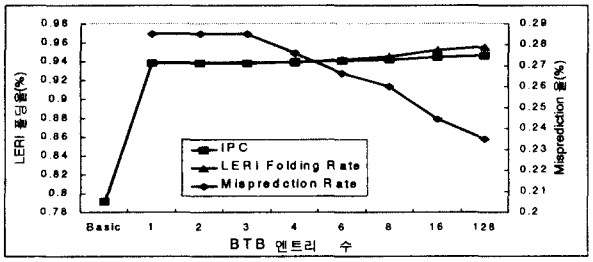

그림 3. BTB 엔트리 개수에 따른 predict-taken기법의 성능 평가

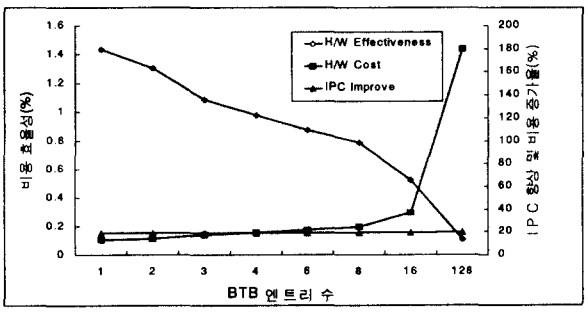

그림 4. BTB 엔트리 개수에 따른 predict-taken기법의 성능 향상율(%), 하드웨어 비용 증가율(%), 비용 효율성(%)

3. 3 1비트 predictor를 갖는 동적 분기 예측 기법

동적 분기 방안들 중 가장 간단한 방법은 이전 분기 명령의 분기 결과를 이용하는 것으로서, 이를 지원하기 위한 하드웨어 비용은 predict-taken방안과 거의 같다. 그림 5에서는 BTB 엔트리의 개수를 변경하면서 분기 예측을 수행하였을 때 성능을 나타낸다. 분기 예측의 정확성은 엔트리가 커지면서 좋아지지만 어느 정도 이상의 성능 향상은 없음을 보여준다. 그림 6은 이 방안을 사용하였을 때 발생하는 성능 향상 비율과 하드웨어 추가 비용 및 비용 효율성을 보여준다.

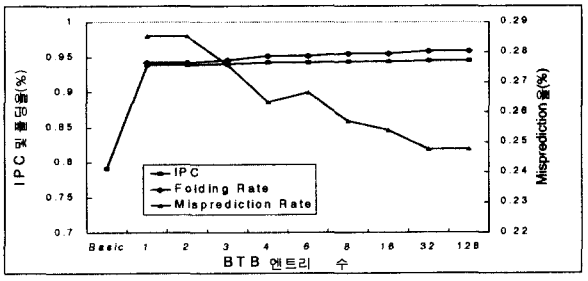

그림 5. BTB 엔트리 개수에 따른 1비트 predictor기법의 성능 향상 및 Misprediction율(%), LERI 폴딩율(%)

3. 4 2비트 predictor를 갖는 동적 분기 예측 기법

동적 분기 예측기로서 bi-modal predictor를 사용하는 경우이며, 이때 BTB의 개수를 변경시키면서 실험을 진행하였다. 이 기법에서 발생하는 추가 하드웨어 비용은 2비트 saturating up-down counter 및 제어 로직이 추가된다. 그림 7은 BTB 엔트리 수에 따른 성능을 보여주며, 그림 8은 이에 따른 성능 향상율과 비용 증가율, 비용 효율성을 보여준다.

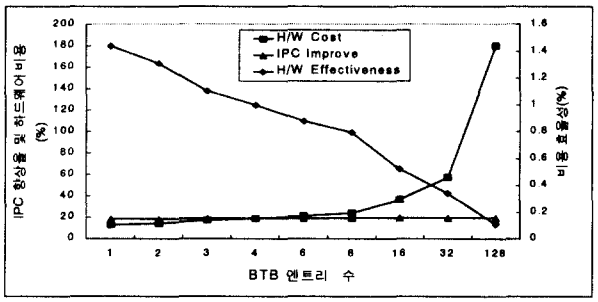

그림 6. BTB 엔트리 개수에 따른 1비트 predictor 기법의 하드웨어 비용 증가(%), 성능 향상(%), 비용 효율성(%)

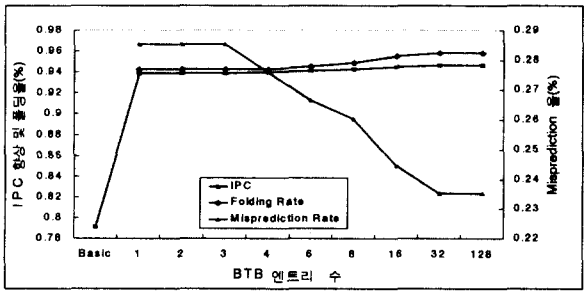

그림 7. BTB 엔트리 개수에 따른 2비트 predictor기법의 성능 향상 및 Misprediction율(%), LERI 폴딩율(%)

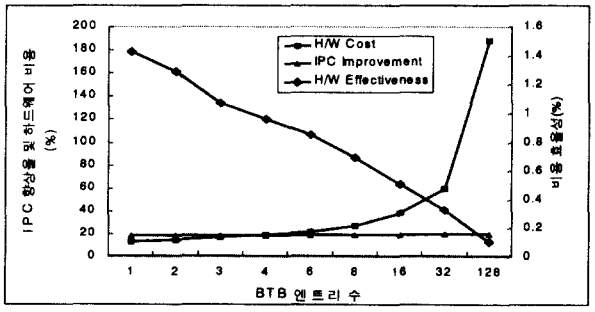

그림 8. BTB 엔트리 개수에 따른 2비트 predictor기법의 하드웨어 비용 증가(%), 성능 향상(%), 비용 효율성(%)

3. 5 결과의 분석

본 절에서는 앞서 제시된 실험 결과로부터 비용 효율성의 측면에서 가장 효과적인 분기 예측 기법을 살펴본다. 그림 9를 보면, IPC의 관점에서는 1비트 predictor/4엔트리 BTB를 이용한 경우가 가장 좋은 결과를 나타내며, 비용 효율성에 있어서는 1엔트리 BTB의 경우가 가장 좋음을 알 수 있다. 따라서, 내장형 마이크로프로세서인 AE32000은 비용 효율성의 측면에서 1개의 BTB 엔트리를 사용하는 것이 바람직한 것으로 판단된다. 또한, EISC 아키텍처의 특성을 고려하여 분기 목적 주소에 나타나는 LERI 명령어를 폴딩 할 수 있도록 BTB를 변경하는 실험이 수행되었으나, 성능 향상을 관찰할 수 없었다. 이는 프로그램 수행 동안 목적 주소에 나타나는 LERI 명령어의 개수가 상대적으로 적었을 뿐 아니라, 이 명령어들이 기 내장된 LERI 명령어 폴딩 유닛에서 효율적으로 처리되었기 때문이다.

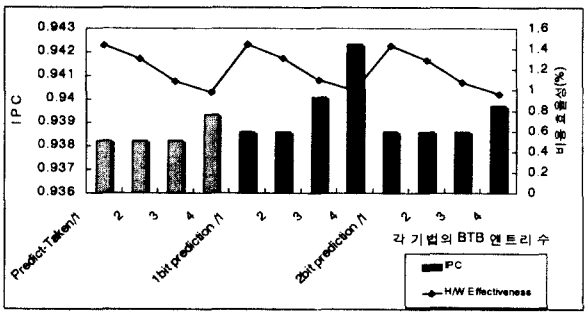

그림 9. 각 기법에 있어서의 IPC와 비용 효율성(%)

4. 결 론

본 보고서에서는 내장형 응용분야에 있어서 사용 가능한 간단한 분기 예측 기법을 AE32000에 적용하여 그 성능을 비교 분석하였으며, 분기 예측이 LERI 명령어의 폴딩에 미치는 영향을 알아보았다. 실험 결과로부터 3개의 LERI 명령어를 폴딩하는 폴딩 유닛과 함께 1비트 predictor를 갖는 1엔트리 BTB를 이용한 분기 예측 기법이 AE32000에서 좋은 비용 효율성을 나타낼 수 있음을 확인하였다. 향후 내장형 프로세서에 적용되는 분기 예측 기법은 내장 환경에서 수행되는 프로그램의 특성이 더욱 잘 반영되도록 연구되어야 할 것이다.

감사의 글

본 연구는 한국반도체연구조합(COSAR) 및 반도체설계교육센터(IDEC)의 지원을 받아 수행되었습니다.

참고문헌

[1] H. Lee, P. Beckett, B. Appelbe, High-performance extendable instruction set computing , *Proc. of ACSAC*, pp.89-94, Jan. 2001.

[2] H.-C. Oh, et al., "AE32000: An Embedded Microprocessor Core", *Proc. of AP-ASIC*, pp.255-258, Aug. 2000.

[3] L.H. Lee, et al., Low-Cost Branch Folding For Embedded Applications With Small Tight Loops, *Proc. of MICRO*, pp.103-111, 1999.

[4] N. Irie, et al., Branch Microarchitecture of SH-5 Embedded Processor , *Proc. of COOL CHIPS IV*, pp.213-222, April 2001.

[5] 김현규, 오형철 EISC용 주기 정밀형 시뮬레이터, *한국군사과학학회 종합학술대회 논문집*, pp.347-350, Aug. 2001.

[6] S. McFarling, J. Hennessy, "Reducing the Cost of Branches", *Proc. of ISCA*, pp.396-403, 1986.