고성능 내장형 마이크로프로세서를 위한 분기예측기의 설계 및 성능평가

이 상 혁, 김 일 관, 최 린
고려대학교 전자공학과
주소 : 서울 성북구 안암동 5가 고려대학교

Branch Predictor Design and Its Performance Evaluation for A High Performance Embedded Microprocessor

Sang-Hyuk Lee, Il Kwan Kim, Lynn Choi
Dept. of Electronics Engineering, Korea University
E-mail : hyukii@korea.ac.kr

Abstract

AE64000 is the 64-bit high-performance microprocessor that ADC Co. Ltd. is developing for an embedded environment. It has a 5-stage pipeline and uses Havard architecture with a separated instruction and data caches. It also provides SIMD-like DSP and FP operation by enabling the 8/16/32/64-bit MAC operation on 64-bit registers. AE64000 processor implements the EISC ISA and uses the instruction folding mechanism (Instruction Folding Unit) that effectively deals with LERI instruction in EISC ISA. But this unit makes branch prediction behavior difficult.

In this paper, we designs a branch predictor optimized for AE64000 pipeline and develops a AE64000 simulator that has cycle-level precision to validate the performance of the designed branch predictor. We makes TAC(Target address cache) and BPT(branch prediction table) seperated for effective branch prediction and uses the BPT(removed indexed) that has no address tags.

I. 서 론

AE64000은 (주)에이디칩스에서 고성능 내장용 환경을 목표로 개발 중에 있는 64비트 마이크로프로세서이다. 5단계 파이프라인으로 되어있고, 명령어 캐쉬와 데이터 캐쉬가 분리된 하바드 아키텍처를 사용하며, 64 비트 레지스터를 분할하여 8/16/32/64 비트의 MAC 연산을 가능하게 함으로써 SIMD 형태의 DSP와 FP 연산을 지원한다. AE64000 프로세서는 EISC ISA를 구현하고 있어 LERI 명령을 효율적으로 처리하기 위해 IFU(Instruction Folding Unit)를 내장하는데 이는 분기예측을 어렵게 한다.

본 논문은 AE64000 파이프라인에 적합한 분기 예측 기를 설계하고 분기 예측을 통한 성능 향상을 평가하기 위해 사이클 수준의 정확성을 갖는 AE64000 시뮬레이터를 제작하였다. 효율적인 분기 예측을 위해 목적 주소 캐쉬(target address cache)와 분기 조건 예측 테이블 (branch prediction table)을 분리하였고 주소 태그 없는 분기 조건 예측 테이블을 구현하였다.

II. AE64000

2.1 AE64000 파이프라인의 전위(Front End)

AE64000은 EISC ISA를 지원한다. EISC란 Extensible Instruction Set Computer의 약자로, 고정된 길이의 명령어에 임의 크기의 operand를 사용할 수 있도록 한다. AE64000은 64비트의 프로세서이지만, 16비트로 고정된 길이의 명령어를 사용한다. 컴파일러에서는 operand를 분할하여 LERI라는 특유의 명령어와 함께 12비트씩 저장한다. 최고 64비트의 길이가 되는 데이터는 5개의 LERI 명령어와 4비트의 immediate 데이터를 사용하는 1 개의 일반 명령어로 처리된다. LERI 명령어 처리에 따른 성능저하를 막기 위해 IFU(Instruction Folding Unit)는 파이프라인의 초기에서 명령어 인출 (fetch)과 동시에 LERI 명령어들을 처리(folding)하고 LERI가 제거된 명령어들을 파이프라인 코어로 보낸다. IFU에서 명령어를 4개씩 인출(fetch)하는데, LERI 제거를 원활하게 하기 위해 IFU내에는 12개의 명령어를 저장하는 버퍼를 가지고 있다. IFU는 LERI 명령어들을 제거하는데 2 clock의 시간이 필요로 한다. 분기 명령이 디코딩(ID) 단계에서 인식이 되면 이미 인출(IF) 단계에 있는 명령어를 포함하여 3 clock의 손실이 있게 된다.

파이프라인 코어에 LERI가 제거된 후 실제 실행하는 명령들의 주소를 갖고 있는 PC와는 별도로, IFU내에는 명령어 캐쉬로부터 인출 주소를 갖고 있는 PPC(pre-PC)가 있다.

2.2 AE64000 분기예측기의 복잡도 (Complexity of Branch Prediction of AE64000)

일반적인 프로세서의 분기예측은 다음과 같이 말할 수 있다.

1.분기가 발생되면 분기명령어의 주소와 분기가 목적하는 주소를 저장한다.
2.명령어의 주소가 PC에서 공급될 때 기존에 저장된 분기명령어의 주소와 비교하여 분기명령의 여부를 확인한다.
3.저장된 분기목적주소의 명령어를 인출한다.

AE64000은 PC에서 공급하는 주소를 이용하여 캐쉬에 접근 하지 않는다. AE64000의 분기실행은 다음과 같다.

1.분기가 발생되면 분기목적주소가 PC에서 계산된다.
2.다음 클럭에 분기목적주소가 PC에서 PPC로 공급되며 동시에 분기발생을 알리는 신호 전송.
3.PPC에서 분기 발생을 인지하면, IFU내의 명령어 버퍼를 모두 삭제한다.
4.다음 클럭에 PPC가 변경되어 분기목적주소를 캐쉬로 공급한다.
5.캐쉬에서 분기목적주소의 명령어를 인출한다.

PPC의 주소는 4개 명령어를 가리키기 때문에 (IFU에서 4개씩 인출하기 위해 PPC주소의 하위 3 비트를 0으로 전환하여 공급한다), 1회의 캐쉬 접근에 대해 4개의 명령어에 대한 분기예측이 필요해진다. 이것이 PPC가 공급하는 주소를 이용하는 분기예측의 장애가 된다.

또한 PPC를 이용한 분기예측은 검증을 위해 예측된 분기목적주소와 실제 분기목적주소를 비교를 하기 어렵다. 분기예측과 분기명령의 실행사이엔 시간적 차이가 있기 때문에 예측된 분기목적주소가 유지되지 않기 때문이다.

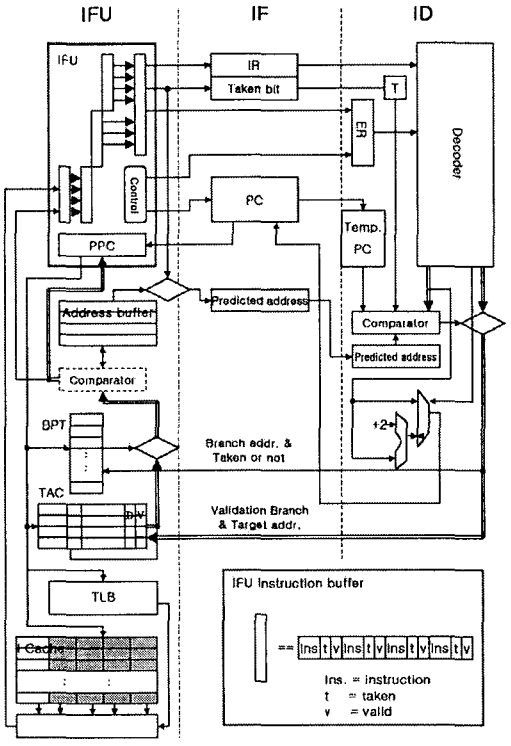

그림 1. Branch Predictor Diagram for AE64000

PC값을 이용한 분기 예측은 분기 예측이 성공하더라도 IFU에서 이미 인출된 명령어들을 플러쉬(flush)하고, IFU에서 소모하는 시간만큼의 분기 손실, 즉 2 clock의 손실을 갖는다.

반면에 PPC를 이용한 분기 예측은 다음 clock에 예측된 주소로부터 인출을 가능하게 함으로써 이러한 손실 없이 효과적인 분기 예측을 가능하게 한다. 따라서 본 논문은 PPC를 이용한 분기 예측을 가정하고 그에 따르는 어려움을 해결하고자 한다.

III. AE64000에 적합한 분기예측

그림 1은 제안된 분기예측기가 추가된 AE64000 파이프라인의 블록 다이어그램이다. AE64000이 내장용이므로 분기목적주소의 저장을 위해 면적의 효율적인 사용이 필요하다. 따라서 분기조건예측기와 목적주소캐쉬를 분리하여 사용, 높은 성능과 면적의 효율성이 높게 하였다.

IFU 단계에서 명령어 캐쉬로부터 인출과 동시에 분기 예측을 위해 BPT (분기예측테이블, 분기조건예측기), TAC (목적주소캐쉬)로 PPC 안의 주소가 보내어 진다. TAC hit가 발생하면 현재 인출되고 있는 4 개의 명령어 중에서 분기 명령이 있음이 확인되고 이 때 BPT의 결과가 taken이면 TAC는 hit된 entry에 저장되어 있는 분기 명령의 목적 주소를 PPC에 제공하여 다음 clock에 이 예측된 주소로부터 명령어를 인출한다. 이 때 IFU는 한번에 4 개의 명령어를 인출하므로 매 clock 마다 최대 4개의 분기 명령어에 해당하는 분기예측을 해야 한다. 하지만 이처럼 4개의 명령어 중에서 2개 이상의 분기 명령이 발생할 확률은 아주 적다. 기존 연구에 따르면 SPECint95에는 평균 4개의 명령어 중 1개의 분기명령이 있고, SPECfp95에는 평균 20개 명령어 중 1개의 분기 명령이 있다. 더욱이 AE64000에서는 LERI 명령의 사용으로 다른 아키텍처와 비교할 때, 캐쉬에서 인출되는 분기명령의 밀도는 작아진다. 그러므로 4개의 명령 중 2번 이상의 분기 명령이 존재하는 경우는 첫 번째 taken된 분기를 예측하고 나머지는 예측 실패를 발생시키는 것이 효율적이다. 이와 같은 방법을 이용하면, 다중 분기 예측 기법(Multiple Branch Prediction, Yhe & Patt)[4]을 사용할 필요는 없다.
또한 [3]에는 1회 분기예측/4개 명령어가 1회 분기예측/1개 명령어에 비하여 성능이 나쁘지 않음을 설명하고 있다.

3.1 분기명령확인

1회의 캐쉬 접근으로 4개의 명령어가 인출된다. 따라서 4개중 몇 번째 명령어가 분기 명령어인지 확인할 필요가 있다. 따라서 분기목적주소캐쉬에 tag를 저장할 때 PPC에서 0으로 전환하여 공급하는 하위 3 비트도 저장하여 정확한 분기 명령의 주소를 저장한다. 이는 분기예측과 함께 IFU에 전송하여 예측된 분기명령 중에서 정확한 분기명령을 구분해 내고, IFU내의 명령어 버퍼에 분기명령임을 표시하여 검증시 분깅예측이 있었음을 알린다.

3.2 비교기(Comparator)

1회 분기예측을 분기예측기에서 예측된 분기 목적 주소를 바탕으로 또 다시 분기예측이 실행되는데, 이때 예측되는 분기 명령이 예전에 예측된 분기 목적 주소의 앞의 주소에 있는 것인지 확인할 필요가 있다.

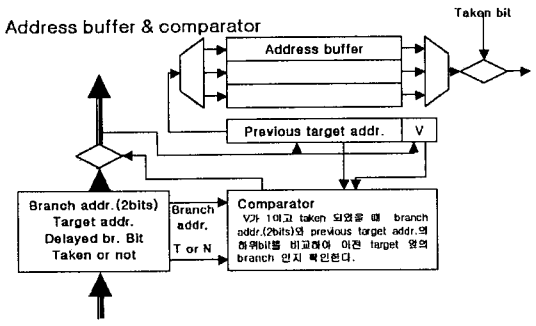

그림 4. Predicted Target Buffer and Comparator

3.3 예측된 분기목적주소 버퍼 (Predicted Target Buffer : Address Buffer)

분기 조건은 분기 명령의 실제 실행으로 검증되어야 한다. 따라서 예측된 분기 목적 주소와 분기 조건은 파이프라인 코어에서 비교되기 위해 IFU 내의 목적 주소 버퍼 (Address Buffer)에 저장된다. 분기명령이 IFU에서 파이프라인 코어로 전송될 때 동시에 전송하는 분기명령임이 예측되

었다는 신호(taken bit)를 받아 예측된 목적주소를 파이프라인으로 공급한다. 이 분기 목적 주소는 디코딩 단계에서 실제 계산된 분기목적 주소와 비교하여 예측된 주소 또는 예측된 조건이 틀린 경우 분기 예측 실패가 되고 인출(IF) 단계와 IFU 안의 명령어 버퍼에 존재하는 명령어들은 삭제(flush)된다. Predicted Target Buffer에는 3개의 목적주소를 저장하는 buffer를 갖는다.

3.4 예측된 목적주소의 검증

분기 명령이 ID단계에서 디코딩되면 분기목적주소가 계산되어진다. 예측된 분기목적주소와 계산된 실제 분기목적주소를 비교하여 분기예측의 성공 여부를 확인한다. 이때 분기예측이 성공하였을 경우는 분기목적주소를 다음 PC값으로 변경하면 안된다. 이미 그 주소의 명령어는 IF 단계에서 실행중이기 때문이다. 따라서 분기예측이 성공하였을 경우는 다음 PC값을 '분기목적주소+2' 로 변경한다. 분기예측의 성공 여부는 분기명령의 주소, 분기목적주소와 함께 BPT, TAC에 전송하여 BPT, TAC 정보를 갱신한다.

IV. 결 론

우리는 본 논문에서 이미 기술한 내용과 같이 명령어를 4개씩 인출하는 AE64000에서 적합한 분기예측기를 제안하고, C language를 이용한 시뮬레이터를 제작, 원활하게 동작함을 확인했다.
다만 부족한 것은 BPT, TAC의 크기에 따른 비용-효율성을 고려하여 최적화된 크기의 분기예측기를 제시하지 못한 것이다. 이는 이미 제작된 시뮬레이터로 실험 중에 있다.

감사의 글

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

참고문헌

[1] T. Y. Yeh and Y. N. Patt, "Alternative Implementations of Two-Level Adaptive Branch Prediction", in Proceedings of the 19th Annual ACM/IEEE International Symposium on Computer Architecture, pp. 124-134, 1992

[2] S. McFarling, "Combining Branch Predictors", DEC WRL Technical Note TN-36, June 1993

[3] Brian K. Bray , M. J. Flynn, "Strategies for branch target buffers". Proceedings of the 24th annual international symposium on Microarchitecture September 1991

[4] T. Y. Yeh, D. Marr, Y. Patt, "Increasing the Instruction Fetch Rate via Multiple Branch Prediction and a Branch Address Cache", in Proceeding of the 1993 International Conference on Supercomputing, pp. 67-76, 1993

[5] John L Hennessy, David A Patterson, "Computer Architecture : A Quantitative Approach 2nd edition", Pressed Morgan Kafmann

[6] D. Burger and T. M. Austin "The SimpleScalar Tool Set, Version 2.0" Computer Architecture News Page13-25 June 1997