1999 年 度 大 韓 電 子 工 學 會 夏 季 綜 合 學 術 大 會 論 文 集 第 22 卷 第 1 號 99/6

명령어 선 인출기의 전력 성능

송영규,오형철
고려대학교 정보공학과
email : {simon,hyeong}@ hard.korea.ac.kr

Power Performance of Instruction Pre-Fetch Unit

Young-Kyu Song, Hyeong-Cheol Oh
Department of Information Engineering, Korea University
email : {simon,hyeong}@ hard.korea.ac.kr

Abstract

In this paper, we investigate the effect of adopting branch-penalty compensation schemes on the power performance of TLBs(Translation Look-aside Buffers) and instruction caches. We found that the double-buffer branch-penalty compensation scheme can reduce the power consumption of the TLBs and the instruction caches considered by up to 14-21.3%. The power consumption is estimated through simulation at the architectural level, using the Kamble/Ghose method

I. 서론

마이크로프로세서를 설계함에 있어서 전력 소모 문제는 시스템의 속도와 면적 이상으로 중요한 문제가 되고 있다. 더구나 향후 시스템 개발 추세가 휴대용 시스템의 점유율이 급속히 증가하는 방향이란 점을 고려할 때, 전력 소모 문제는 더욱 중요성을 가지게 된다. 한편, 현재의 마이크로프로세서에서 전력 소비의 많은 부분을 책임지고 있는 소자중의 하나가 메모리이므로 메모리 접근 횟수를 줄이는 것은 전력 소비 측면에서 매우 중요하다.

명령어 선 인출기는 이중 명령어 버퍼를 사용하여 분기 명령어에 대해 선택된 경로와 그렇지 않은 경로에 해당되는 명령어를 모두 선 인출하여 예측이 틀렸을 경우 새로운 타겟을 요청함에 따른 손실을 선 인출 해 놓은 다른 경로의 명령어를 수행함으로써 줄일 수가 있다.

본 논문에서는 이러한 분기 손실 보상 기법의 사용이 명령어 TLB 와 명령어 캐쉬의 전력 소모에 미치는 영향을 다룬다.

전력 분석을 위해 사용된 TLB 는 전역 사상(Fully-associative Mapping)방식으로 설계를 하였으며, 명령어 캐쉬의 경우 직접 사상(Direct Mapping) 방식으로 설계를 하였다. 전역 사상 방식은 다양한 크기의 페이지를 지원하기가 쉽고 히트 비율도 직접 사상방식보다 상대적으로 좋다. 그러나 CAM(Content Addressable Memory)셀을 사용하므로 매 사이클마다 명령어가 발생한다고 가정하였을 경우 매번 CAM 의 모든 엔트리를 다 접근해야 하기 때문에 전력 소비면에서 직접 사상 방식보다 떨어지게 된다. 본 논문에서는 분기 예측 실패에 따른 손실 보상을 위한 이중 버퍼 기법[6]이 전역 사상 방식의 명령어 TLB 와 직접 사상 방식의 명령어 캐쉬의 전력 소비에 미치는 영향을 구조 수준(Architectural Level)[1]에서 고찰하였다.

본 논문에서 사용하는 MMU 와 명령어 페처는 Summit 사의 Visual_HDL Tool 을 사용하여 Verilog_HDL 로 설계되었으며 다중 프로세서인 RAPTOR[2]의 MMU 와 명령어 페치 유닛 설계에 사용하여 성능을 검증하였다.

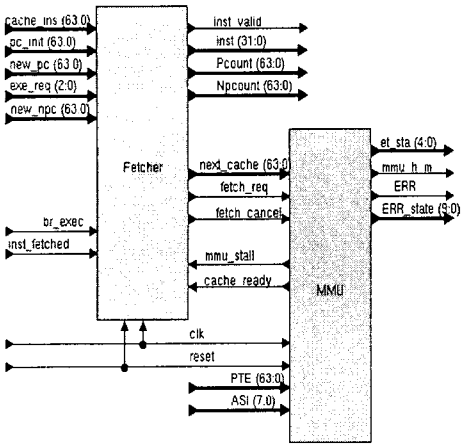
II. 명령어 선 인출기의 구조

1. RAPTOR 의 구조

RAPTOR[2]는 독립된 4 개의 프로세서 유닛으

로 구성된 쓰레드 수준의 단일 칩 프로세서로서 SPARC V9[5]명령어 셋 구조를 가지며 64 비트 정수 및 부동 소수점 연산을 한다. 그리고 크기가 비교적 작은 내장 1 차 캐쉬와 대용량의 외부 2 차 캐쉬로 구성된 다중 캐쉬 구조를 갖는다.

RAPTOR 의 파이프라인 중 전 처리단계에 해당하는 명령어 선 인출기는 명령어 페처, MMU 블록으로 구성되었다. 명령어 선 인출기는 수행될 명령어를 MMU 로 요청하여 명령어 캐쉬로부터 해당 명령어들을 받아 명령어 버퍼에 저장한다. 그림 1 은 RAPTOR 에서 사용되는 명령어 선 인출기의 블록도이다.



[그림 1] 명령어 선 인출기의 블록도

2. MMU 의 구조

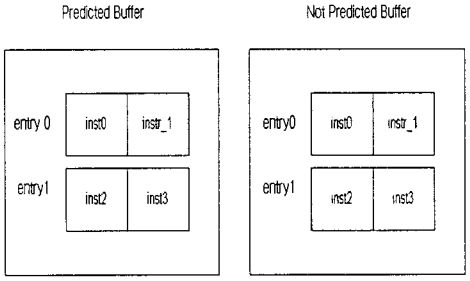
프로세서에서 발생시키는 64 비트 가상 주소를 실제 주소로 변환 해 주는 MMU 의 구조와 기능은 다음과 같다.

- Ultra-SPARC[5]의 MMU 를 모델로 설계하였다.
- 명령어 TLB 는 64 엔트리를 갖는 전역 사상방식으로 구현하였다.
- 가상 주소를 특정 프로세서로부터 받아 실제 주소로 변환한다.
- 메모리 액세스 제한 및 체크를 통하여 메모리 영역 보호(Memory Protection)와 각 프로세서의 보안 유지 기능을 담당한다.
- 메모리 할당의 효율을 높이기 위한 다중 계층 페이지 테이블을 지원한다.
- 페이지 오류 발생시에 효과적이고 빠른 테이블 워크를 지원한다.

3. 명령어 페처의 구조

Raptor 의 명령어 페치 유닛에서는 그림 2 와 같이 이중 명령어 버퍼를 이용하여 브랜치 폴딩 기법을 구현하였다.

두 개의 명령어 버퍼를 사용함으로써 한 개의 조건 분기 명령어에 대해서 예측된 경로와 그렇지 않은 경로에 두 명령어 버퍼를 할당하고, 각각의 경로에 해당하는 명령어들을 선 인출 해둠으로써 분기 예측이 실패하였을 경우 예측되지 않았던 경로에 해당된 명령어들을 수행하고, 다음 명령어들을 요청함으로써 분기 손실을 줄일 수 있다. 한 개의 조건 분기 명령어에 대해서만 명령어 버퍼를 할당하여 처리하므로, 연속된 조건 분기 명령어의 경우 두 번째 명령어의 정보는 저장하였다가 첫 번째 분기 명령어가 처리된 후 그 결과에 따라 두 번째 분기 명령어를 수행하도록 하였다. 명령어 버퍼의 수를 증가함으로써 처리 가능 조건 분기 명령어의 수를 늘리거나, 엔트리 내의 처리 명령어 수를 증가 시켜 가능 유효 명령어의 수를 늘림으로써 분기 손실의 감소 효과가 커질 수 있다.



[그림 2] 이중 명령어 버퍼를 사용한 분기 폴딩

명령어 페처는 수행될 명령어들을 명령어 캐쉬로부터 프리페치하여 명령어 버퍼에 저장하여 두며, 명령어들을 프리디코드 한다. 프로그램 수행의 순서를 바꾸는 명령어들(예: 분기 명령어 등)을 구별하여 해당 타겟 명령어들을 프리페치하여 연속적인 명령어 수행이 이루어지게 한다. 두 개의 명령어 버퍼를 두어 순차적 명령어 블록인 경우 한 개의 명령어 버퍼를 사용하여 프리페

치 해 두며, 분기 명령어의 타겟 주소에 해당되는 명령어는 또 다른 한 버퍼에 프리페치하여, 예측된 경로에 해당되는 명령어들을 수행하며 분기 명령어의 처리 결과에 따라 예측이 틀렸을 경우 다른 버퍼의 명령어를 수행함으로써 분기 손실을 줄인다.

III. 전력 평가 방법 및 실험

1. 전력 평가 방법

TLB도 페이지 테이블의 엔트리를 저장하는 캐쉬 구조를 가지므로 Raptor 시뮬레이터를 이용하여 아래의 Kamble/Ghose 방법[3]을 사용하여 구조 수준에서 실험을 하였다.

CMOS 공정에서 소모되는 에너지는 다음 식(1)과 같이 부하 정전용량이 소자들에 대한 충전/방전 전이에 대한 식으로 표현 할 수가 있다.

$$E = 0.5 \cdot C \cdot V^2 \quad (1)$$

여기서 C는 부하 정전용량이고 V는 소자에 공급되는 전압이다. 그리고 캐쉬 전체에 소모되는 전력 소비는 다음 식(2)과 같이 구할 수가 있다.

$$E_{total} = E_{bit} + E_{word} + E_{output} + E_{ain} \quad (2)$$

각각의 요소에 대한 설명은 다음과 같다.

- E_{bit}(Energy Dissipated in Bit-Lines) : 비트 라인(Bit-Line)에서 소모되는 에너지로서 Precharging, 읽기/쓰기 동안에 소모되는 에너지이다.

$$E_{bit} = 0.5V^2 \cdot [N_{bit.pr} \cdot C_{bit.pr} + N_{bit.w} \cdot C_{bit.r/w} + N_{bit.r} \cdot C_{bit.r/w} + m \cdot (8 \cdot L + T + St) \cdot CA \cdot (C_{q.qpa} + C_{q.qpb} + C_{q.p})] \quad (3)$$

- E_{word}(Energy Dissipated in Word-Lines) : Word-Line에서 소모되는 에너지로서 하나의 열이 선택될 때 필요한 에너지이다..

$$E_{word} = V^2 \cdot CA \cdot m \cdot (L \cdot 8 + T + St) \cdot (2 \cdot C_{q.ql} + C_{wordwire}) \quad (4)$$

- E_{output}(Energy Dissipated Driving External Buses) : 데이터를 버스로 송/수신 할 때나 미스 시 주소를 하위 메모리로 보낼 때 소모되는 에너지이다.

$$E_{output} = E_{aoutput} + E_{doutput} \quad (5)$$

$$E_{aoutput} = 0.5 \cdot V^2 \cdot (N_{out.a2m} \cdot C_{out.a2m} + N_{out.a2c} \cdot C_{out.a2c}) \quad (6)$$

$$E_{doutput} = 0.5 \cdot V^2 \cdot (N_{out.d2m} \cdot C_{out.d2m} + N_{out.d2c} \cdot C_{out.d2c}) \quad (7)$$

- E_{ain} (Energy dissipated in the address input lines) : 주소(캐쉬의 열) 해독기의 입력 단에서 소모되는 에너지이다.

$$E_{ainn} = 0.5 \cdot V^2 \cdot N_{ainput} \cdot [(m+1) \cdot 2 \cdot S \cdot C_{in,dec} + C_{awire} \quad (8)$$

여기서 TLB의 경우 전역 사상 방식의 구조이므로 주소 입력 라인에서 소모되는 에너지(E_{ain})는 없게 된다. 사용된 문자의 의미는 다음과 같다.

m : 연산도 L : 라인의 크기(byte)
T : 태그의 크기 D : 전체 용량
CA : 총 캐쉬 접근 횟수
St : 블록 당 상태 비트의 수)
$C_{x,y}$: 부하 정전용량(Load Capacitances)
$N_{x,y}$: 전이 횟수(Number of Transition

정전용량 계수(Capacitive Coefficient)들의 값은 Wilton/Jouppi[4]에서 사용한 0.8마이크론 CMOS 공정 파라미터(Parameters)을 사용하였고, 시뮬레이터를 통한 명령어 TLB와 명령어 캐쉬의 히트/미스 수, 메모리 접근 수 등을 [3]에서 제시된 방법으로 각각의 식(3)~식(8)에 필요한 전이 수를 얻었다. 나머지 파라미터들은 [4]을 참조하였다.

2. 실험 결과

단일 버퍼 기법과 본 논문에서 제안하는 이중 버퍼 기법을 사용하여 전력 성능에 대한 실험은 TLB 엔트리의 크기에 따른 전력 성능 비교와 명령어 캐쉬에 대한 전력 성능 비교를 수행하고 있다.

전력 분석을 위해 사용된 명령어 TLB와 명령어 캐쉬 그리고 명령어 버퍼의 사양은 다음과 같다.

- 32, 64, 128 엔트리를 갖는 전역 사상(Fully-Associative)방식의 명령어 TLB
- 32bytes 의 라인 크기를 갖는 16Kbytes 의 직접 사상(Direct Mapping)방식의 명령어 캐쉬
- 64 비트의 엔트리를 갖는 명령어 버퍼

현재 SPEC CPU95 벤치마크 프로그램 중 데이터 베이스 프로그램인 Vortex와 C 언어로 작성된 작은 규모의 응용 프로그램인 Quicksort를 대상으로 실험을 수행하였으며, Quicksort 프로그램은

2 회 이상 반복 수행하였다. 그 실험 결과는 아래 표 1 과 표 2 에 보였다.

Single Buffer			
벤치마크	32entry	64entry	128entry
Vortex	0.602	0.726	1.380
Quicksort	0.028	0.056	0.081

(a) Single Buffer

Double Buffer			
벤치마크	32entry	64entry	128entry
Vortex	0.592	0.697	1.027
Quicksort	0.023	0.034	0.076

(b) Double Buffer

[표 1] 명령어 TLB 의 전력 성능

벤치마크	Single Buffer	Double Buffer
Vortex	0.901	0.835
Quicksort	0.081	0.073

[표 2] 명령어 캐쉬의 전력 성능

표에서 볼 수 있는 바와 같이, TLB 의 전력 소모는 이중 명령어 버퍼기법이 단일 명령어 기법보다 Vortex 의 경우 15.3%, Quicksort 는 21.3%정도 적게 소모되었고 명령어 캐쉬의 경우도 Vortex 는 7%, Quicksort 에서는 10%정도 적게 소모되었음을 알 수 있고, 이중 명령어 버퍼기법을 사용 할 경우 두 가지 프로그램에 대한 TLB 와 캐쉬의 전체적인 전력 성능은 Vortex 는 약 6%, Quicksort 는 14%의 전력 향상을 보였다.

실험 결과 중 단일 명령어 버퍼기법의 경우 32 엔트리를 갖는 TLB 의 전력 성능은 이중 명령어 버퍼기법의 전력 성능과 비교했을 때 거의 차이가 없으며 TLB 의 엔트리 수가 증가할수록 전력을 더 많이 소모하는 것을 알 수가 있다. 이것은 TLB 가 전역 사상방식으로 설계되었기 때문에 명령어 페쳐에서 주소를 발생시킬 때마다 CAM 의 모든 엔트리를 다 접근해야 하므로 엔트리 수가 증가 할수록 CAM 의 비트라인(Bit Line)에서 소모되는 전력이 증가하게 된다.

V. 결론 및 향후 계획

명령어 TLB 와 명령어 캐쉬에 대한 이중 명령어 버퍼 기법과 단일 명령어 버퍼 기법의 전력 성능 비교 분석 결과 이중 명령어 버퍼 기법이 전체적으로 약 10%의 전력 소모를 절약 할 수가 있다. 전력 성능 면 이외에도 수행 시간 비교 결과 약 15%정도의 성능 향상이 있다[6]. 이와 같은 결과는 분기 예측이 틀렸을 경우 미리 인출해 놓은 다른 경로의 명령어를 수행함으로써 단일 버퍼에서 발생할 수 있는 분기 예측 실패에 따른 손실을 줄일 수가 있기 때문이다.

본 실험에서 사용 한 Kamble/Ghose 방법의 전력 분석 오차 율은 2%내외로 알려졌다[3].현재보다 더 정확한 분석을 위해, 여러 개의 벤치마크 프로그램들을 대상으로 실험을 수행 중이며, 그 결과는 본 논문의 최종 본에 추가될 것이다.

참고 문헌

[1] R.Y.Chen , R.M.Owens , M.J.Irwin. "Validation of an Architectural Level Power Analysis Technique", Proc.of DAC '98, pp.242-245

[2] 이상원, 김영우, 오형철, 김수원, 한우종, 윤석한. "단일 칩 다중프로세서의 설계", 1998 년도 대한전자공학회 추계학술대회, pp.751-754, 1998. 11.

[3] Kamble.M.B , Ghose.K. "Analytical Energy Dissipation Models for Low Power Caches", Proc. of ISCA International Symposium on Computer Architecture , pp.143-148, August 1997.

[4] Wilton .S.E, Jouppi.N. "An Enhanced Access and Cycle Time Model for On-chip Caches", DEC WRL Research Report 93/5, July 1994.

[5] David L., Weaver, Tom Germond, *The SPARC Architecture Manual Ver.9*, SPARC International Inc.

[6] 이성권 ,오형철. "RAPTOR 의 명령어 페치 유닛 설계", 1998 년도 대한전자공학회 추계학술대회, pp.767-770, 1998.11.