다중처리형 마이크로프로세서 미세구조 시뮬레이터

박 경, 한우종
한국전자통신연구원 컴퓨터시스템 연구부
305-350 대전시 유성구 가정동 161번지
{kyong, wjhan}@computer.etri.re.kr

Microarchitecture Simulator for On-Chip Multiprocessor Microprocessor

Kyoung Park, Woo-Jong Hahn
Computer System Department, ETRI
161 Kajong-Dong, Yusong-Gu, Taejon, 305-600, Korea
{kyong, wjhan}@computer.etri.re.kr

Abstract

Microarchitecture simulator is an important tool to verify and optimize the microarchitecture of a new microprocessor. Moreover, it can be use as a performance simulator to estimate the target microprocessor's performance. And system software designers can use it as a software developing environment.

This paper describes a "microarchitecture simulator for on-chip Multiprocessor microprocessor". It is a program-driven and cycle-based simulator that can execute simultaneous mutithreading benchmarks. We verified the microarchitecture of a new on-chip multiprocessor microprocessor with it and did performance simulations to estimate the performance of the on-chip multiprocessor microprocessor.

I. 서론

일반적으로 마이크로프로세서 개발 순기는 미세구조 설계와 반도체 설계로 구분할 수 있다. 미세구조는 명령어 세트 및 마이크로프로세서 내부 구조와 기능을 정의하는 것으로 이를 바탕으로 반도체 설계가 진행된다[1]. 반도체 설계는 설계 및 검증을 위한 상용 개발 환경(CAD)이 잘 구축되어 설계 및 검증에 다양한 도구를 사용할 수 있다. 하지만 미세구조 설계는 설계 및 검증에 사용할 만 개발 환경이 없는 현실이다. 미세구조는 명령어 세트 및 마이크로프로세서 내부 구조와 기능을 정의하는 것으로 이를 바탕으로 반도체 설계가 진행된다. 따라서 미세구조 설계 및 검증 없이 반도체 설계를 진행하는 것은 불가능하다.

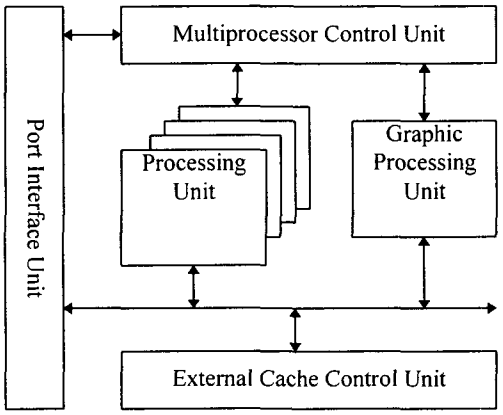
대부분의 마이크로프로세서 개발회사들은 고유의 미세구조 개발환경을 구축하고 있으며, 이를 사용하여 명령어 세트의 무결성을 검증하고 미세구조 검증 및 최적화에 사용하고 있다. 미세구조 개발환경은 주로 미세구조를 모델링한 시뮬레이터 환경을 사용하고 있다.

미세구조 시뮬레이터는 미세구조 설계 및 검증, 미세구조 최적화 등의 목적외에 컴파일러, 운영체제와 같은 시스템 프로그램 개발 플랫포옴으로도 사용되며 마이크로프로세서의 성능 평가에도 사용되어 진다[2].

본 논문에서는 한국전자통신연구원에서 수행한 다중처리형 마이크로프로세서 개발사업에서, 다중처리형 마이크로프로세서 미세구조 개발환경으로 사용된 미세구조 시뮬레이터에 대하여 기술한다.

서론에 이여 2 장에서는 개발대상인 다중처리형 마이크로프로세서에 대하여 기술하고 3 장에서는 미세구조 시뮬레이터의 구조 및 기능에 대하여 기술한다. 4 장에서는 미세구조 시뮬레이터의 실행 결과를 기술하고 5 장에서 결론 및 향후 연구에 대하여 기술한다.

II. 다중처리형 마이크로프로세서



<그림 1> 다중처리형 마이크로프로세서

다중처리형 마이크로프로세서[3,4]는 프로그램 카운터와 연산 장치를 갖는 프로세싱 코어 4 개와 멀티미디어

데이터 처리를 위한 그래픽 처리기를 하나의 마이크로프로세서로 집적한 것으로 <그림 1>과 같은 구조이다.

2.1 프로세싱 유니트

프로세싱 유니트는 SPARC V9 명령어 세트[5]를 기본으로 설계된 명령어를 처리하며 비순차 실행 기능[1]과 분기예측 기능[1]을 포함한 2-way 수퍼스칼라 구조의 RISC 프로세서이다. 하버드 구조의 내부 캐쉬와 MMU(Memory Management Unit), 프로그램 카운터를 갖는 연산 파이프라인, 부동소수점 연산장치로 구성된다. 64 비트 데이터와 가상 어드레스를 처리한다.

2.2 그래픽 프로세싱 유니트

그래픽 프로세싱 유니트는 멀티미디어 데이타 처리 성능 향상을 위하여 내장되는 하드웨어로서 그래픽 전용 명령어를 수행한다. 4 개의 프로세싱 유니트에 의해서 공유되나, 한 순간에는 하나의 프로세싱 유니트와 동기 되어 명령어 흐름에 포함 된 그래픽 명령어를 처리한다. 그래픽 처리를 위한 연산기와 그래픽 데이타를 위한 레지스터 파일로 구성된다.

그래픽 데이타의 처리는 각 화소 정보 단위로 동시에 연산할 수 있는 SIMD(Single Instruction Multiple Data) 형태로 이루어 진다. 아울러서 MPEG 이나 화상 회의 지원을 위한 전용 하드웨어를 내장한다.

2.3 다중처리 제어 유니트

다중처리 마이크로프로세서의 기본 설계 개념은 프로세싱 엘리먼트 수준의 병렬성 지원이다. 따라서 4 개의 프로세싱 유니트가 다중 처리 시스템으로 동작할 수 있도록 부가적인 기능을 지원하는 하드웨어가 요구된다. 다중처리 제어 유니트는 클럭을 분배하고, 4 개의 프로세싱 유니트의 초기화 및 외부 인터럽트의 분배 그리고 프로세싱 유니트 간 동기화를 지원한다. 특히 프로세싱 유니트간 동기화 지원은 다중 처리 시스템 구현의 필수 기능으로 소프트웨어에서 빠르고 쉽게 접근 가능한 동기화 자원을 제공한다.

2.4 외부 캐쉬 제어 유니트

4 개의 프로세싱 유니트에서 발생하는 메모리 접근을 빠르게 하기 위하여 프로세서 외부에 명령어/데이타 혼합형 외부 캐쉬를 구성한다. 외부 캐쉬 제어 유니트는 외부 캐쉬의 상태 및 데이타 일관성을 제어한다. 4 개의 프로세싱 유니트에 내장된 내부 캐쉬와 다른 프로세서들의 캐쉬와 메모리간에 일관성을 유지해야 하며, 이를 위하여 시스템 인터페이스와 내부 버스를 동시에 스누핑을 수행한다.

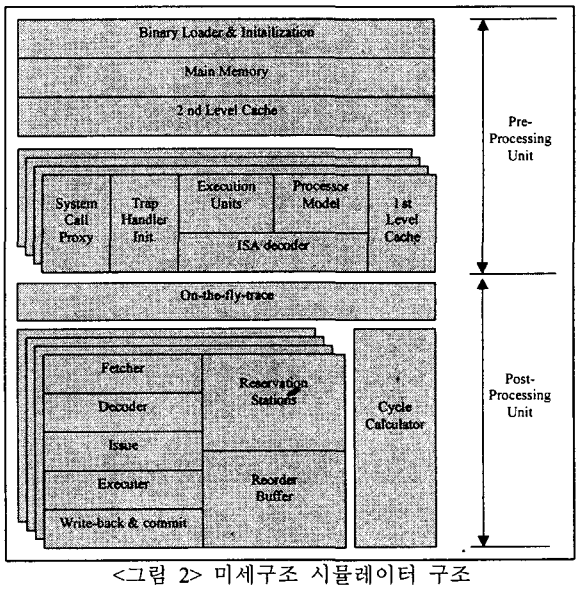
2.5 시스템 인터페이스 유니트

시스템 인터페이스 유니트는 다중처리 마이크로프로세서 내부에서 발생하는 메모리 접근 요구, 입출력 장치 접근 요구, 인터럽트 및 기타 유틸리티 신호선의 사용 요구를 처리한다. 또한 부가적인 외부 로직 없이 다중 처리 시스템을 구성할 수 있는 MP-ready 형태의 인터페이스를 제공한다.

III. 미세구조 시뮬레이터

미세구조 시뮬레이터는 다중처리형 마이크로프로세서 미세구조 설계 및 검증과 최적화를 위하여 개발되었으며, 아울러서 최적화 캄파일러 개발 및 프로그램 환경 개발에도 사용되었다.

미세구조 시뮬레이터는 컴파일러에 의해서 생성된 이진 실행 파일을 직접 입력으로 받는 프로그램 구동형으로 개발되었으며, 싸이클 단위로 마이크로프로세서 내부 상태를 관리하는 싸이클 기반형 시뮬레이터이기도 하다. <그림 2>는 미세구조 시뮬레이터의 구조이다.



<그림 2> 미세구조 시뮬레이터 구조

미세구조 시뮬레이터는 크게 전처리부와 후처리부로 구성되어 있으며, 전처리부에서 생성한 실시간 트래이스 정보를 사용하여 후처리부가 파이프라인 모델을 시뮬레이션 한다.

3.1 전처리부

전처리부는 프로그램 구동형 시뮬레이터 형태로 구성되어 있다. 명령어를 수행하는 실행 유니트와 레지스터 파일을 비롯해서 프로세서 내부 상태를 모델링한 프로세서 모델, 캐쉬 및 메모리 모델, 그리고 시스템 콜 및 트랩 처리를 위한 부분으로 구성되어 있다.

이진 실행파일을 입력으로 받기위해서 ELF 파일 해독기를 내장하고 있으며, 해독된 ELF 파일은 코드 부분과 데이터 부분으로 분할되어 메모리 모델에 적재된다.

프로세서 모델은 메모리 모델을 접근하여 명령어를 패치하고 패치된 명령어는 명령어 디코더와 실행 유니트에 의해서 처리진다. 실행 유니트는 실행 결과를 프로세

서 모델에 반영하여 레지스터 파일 및 기타 프로세서 내부 제어 레지스터 및 상태 레지스터를 갱신한다. 실행 유니트 및 프로세서 모델에 의해서 발생되는 메모리 접근 요구는 캐쉬 모델을 통하여 이루어 지며, 이때 내부 캐쉬 및 외부 캐쉬의 적중 여부가 관리되어 진다.

전처리부는 실제 입력된 이진 파일을 수행하는 부분으로 매 명령어 처리시 마다 실시간 트래이스 정보를 생성하여 후처리부로 전달한다. 실시간 트래이스는 명령어 종류, 프로그램 카운터, 오퍼랜드 형태, 캐쉬 적중 여부, 실행 처리 시간, 분기 발생 여부 등 파이프라인 제어에 필요한 정보를 모두 포함하고 있다.

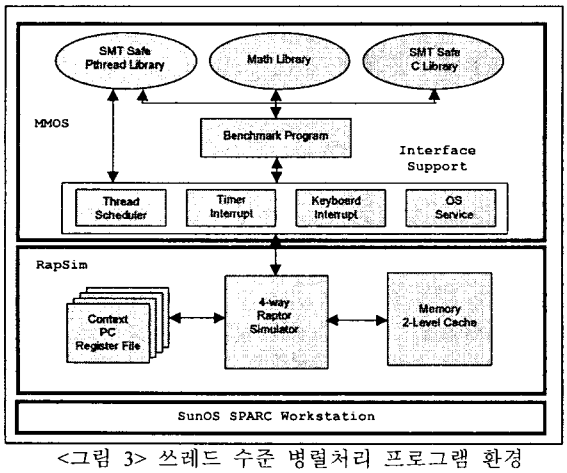
3.2 후처리부

후처리부는 전처리부에 의해서 생성된 트래이스를 사용하여 실제 파이프라인의 효율을 싸이클 단위로 계산하는 부분이다. 비순차 실행 및 분기 예측 기능을 갖는 2-way 수퍼스칼라 5단 파이프라인을 모델링한 것으로 재정렬 버퍼와 레저베이션 스테이션을 사용한다.

전처리부에서 발생한 실시간 트래이스를 관리하면서 순서대로 명령어를 패치하여 해당 기능 유니트의 레저베이션 스테이션으로 이슈한다. 동시에 재정렬 버퍼에 이슈한 명령어를 등록한다. 이슈된 명령어는 명령어간 종속성 및 하드웨어 가용도에 따라서 비순차적으로 처리되며, 처리된 명령어는 재정렬 버퍼를 거쳐서 레지스터 파일 갱신 후 제거된다.

후처리부는 싸이클 단위로 파이프라인 모델을 제어하며, 동시에 성능 평가를 위하여 총 수행 싸이클 시간 및 실행 명령어 수와 같은 성능 인자를 관리한다.

3.3 다중처리 모델



<그림 3> 쓰레드 수준 병렬처리 프로그램 환경

전처리부와 후처리부로 구성되는 하나의 프로세서 모델을 4개로 확장하여 다중처리형 마이크로프로세서 미세구조 시뮬레이터를 제작하였다. 다중처리형 마이크로프로세서 미세구조 시뮬레이터는 4개의 프로세서 모델이 2차 캐쉬와 메모리 모델을 공유하는 2차 캐쉬 공유형 다중처리 프로세서이다.

4개의 프로세서를 효율적으로 사용하기 위하여 쓰레드 수준 병렬처리[6] 프로그램 환경을 사용한다. 쓰레드 수준 병렬처리 프로그램 환경은 POSIX 쓰레드 라이브러리를 수정하여 개발하였다. 또한 GNU C 라이브러리 및 MATH 라이브러리에서도 쓰레드간 동기화 문제를 해결하기 위하여 라이브러리 일부를 수정하였다. 쓰레드를 프로세서 모델에 할당하고 관리하기 위하여 별도의 쓰레드 관리 라이브러리를 제작하였다. <그림 3>은 다중처리형 마이크로프로세서 미세구조 시뮬레이터와 쓰레드 수준 병렬처리를 위한 프로그램 환경을 보여주는 그림이다.

3.4 벤치마크 프로그램

다중처리형 마이크로프로세서 미세구조 검증을 위하여 사용된 벤치마크는 SPLASH 벤치마크 세트[7]에서 사용한 LU, FFT, MP3D 와 자체적으로 작성한 매트릭스 곱셈과 가우스 소거법 프로그램이다.

벤치마크 프로그램은 다중처리 프로그램 환경의 라이브러리를 사용하여 다중쓰레드 프로그램 모델을 갖도록 제작되었으며, 동일한 라이브러리와 컴파일시에 링크되어 최종적인 이진 파일을 생성한다.

IV. 미세구조 시뮬레이터 실행 및 결과

다중처리형 마이크로프로세서 미세구조 시뮬레이터 및 쓰레드 수준 병렬처리 프로그램 환경은 C 언어를 사용하여 구현되었으며, Solaris 2.5 이상을 운영체제로 사용하는 SUN 워크스테이션에서 동작한다.

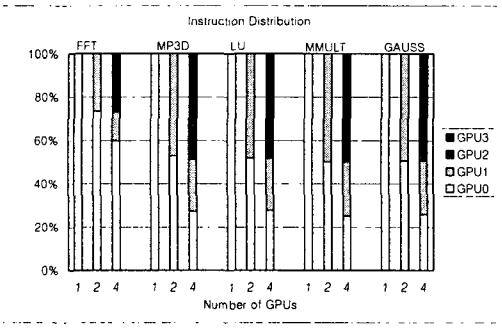
<표 1> 시뮬레이션 파라미터

Feature	Default Value
Number of GPUs (*P*)	4 (1, 2, 4)
GPU issue width	2
1st cache Size	16 Kbyte I-cache / 32 bytes line 16 Kbyte D-cache / 32 bytes line
2nd cache size	4 Mbyte / 32 bytes line
Write update policy	1st cache to 2nd cache : write through 2nd cache to Main memory : write back
1st cache access latency	1 cycle
2nd cache access latency	4 cycle
Main Memory access latency	10 cycle
Instruction Execution Latency	Integer ALU = 1 cycle Integer Multiply = 4 ~ 34 cycle Integer Division = 36 (single), 68(double) cycle Load/Store = 1 cycle Control Transfer = 1 cycle FP Addition/Subtraction = 1 cycle FP Multiply = 4 cycle FP Division = 12(single), 22(double) cycle

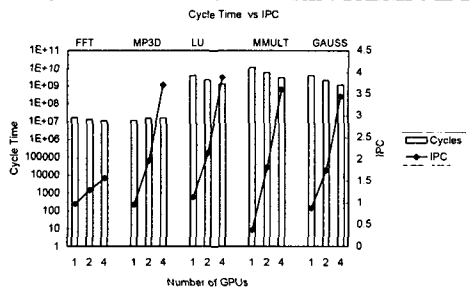
벤치마크 프로그램을 컴파일하여 생성된 이진 파일

을 입력으로 동작하며, 시뮬레이션을 통하여 캐쉬 효율, 총 수행 명령어 수, 총 수행 싸이클 수, 명령어 분포 등과 같은 성능 인자를 관리하고 추출한다.

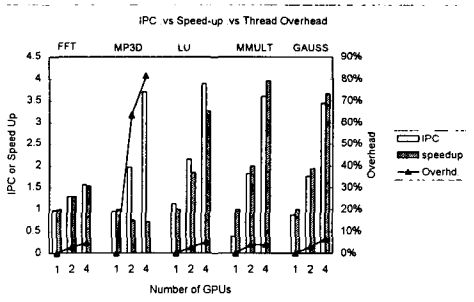
<표 1>은 다중처리형 마이크로프로세서 미세구조 시뮬레이터 실행을 위한 시뮬레이션 파라미터를 정리한 표이다. <그림 4>, <그림 5>, <그림 6> 은 <표 1>의 파라미터를 가지고 시뮬레이션한 결과를 보여주는 그래프이다.



<그림 4> 쓰레드수준 병렬처리 효과



<그림 5> 싸이클 시간 vs. IPC



<그림 6> IPC, Speed-up vs. 쓰레드 오버헤드

<그림 4>는 다중처리형 마이크로프로세서에서 쓰레드 수준 병렬처리 효과를 보여주는 그래프로 프로세서 개수가 증가함에 따라 명령어가 각 프로세서에 균등하게 분배되는 효과를 보여주고 있다. <그림 5>는 쓰레드 수준 병렬처리시에 프로세서 개수가 증가함에 따라 실행 싸이클은 감소하고 IPC(Instructions Per Cycle)은 증가하는 것을 보여준다. <그림 6>은 쓰레드 수준 병렬처리에 있어서 쓰레드 처리를 위한 오버헤드를 측정한 것으로 쓰레드 오버헤드가 적을수록 프로세서 개수에 비례하여 Speed-up 이 선형에 가깝게 증가하는 것을 보여준다.

V. 결론 및 향후 계획

본 논문에서는 다중처리형 마이크로프로세서 개발에 있어서 미세구조 설계 및 검증을 위하여 사용된 미세구조 시뮬레이터의 구조 및 시뮬레이션 결과에 대하여 기술하였다. 개발된 미세구조 시뮬레이터는 프로그램 구동형 싸이클 기반형 시뮬레이터로 초기 미세구조 설계 및 검증에 사용되었으며, 미세구조 최적화 및 성능 평가에도 사용되었다. 또한 다중처리 지원을 위한 쓰레드 수준 병렬처리 프로그램 환경을 동시에 개발하여 다중처리형 마이크로프로세서에 쓰레드 수준 병렬처리를 적용시켜 보았다.

향후에는 현재까지 개발된 미세구조 시뮬레이터를 확장하여 시스템 시뮬레이션을 수행할 수 있는 시뮬레이터로 사용할 예정이며, 특히 소규모 운영체제를 시뮬레이터에 이식하는 작업을 수행할 예정이다.

참고문헌

[1] M. Slater, "The Microprocessor Today," *IEEE Micro*, Vol. 16, No. 6, pp. 32-45, 1996.

[2] D. Burger, and T. M. Austin, "The SimpleScalar Tool Set, Version 2.0", Univ. of Wisconsin-Madison Computer Sciences Dept. TR#1342, June, 1997.

[3] L. Hammond, et al., "A Single-Chip Multiprocessor," *IEEE Computer*, Vol. 30, No. 9, PP. 79-85, 1997.

[4] W. J. Hahn, et al., "A study on logic design and architecture simulator design of a new on-chip multiprocessor", Proc. of IEEE TENCON'99, Vol. 1, pp.475-478, September 1999.

[5] SPARC International, Inc., *The SPARC Architecture Manual version 9*, 1994.

[6] S. Egger, et al., "Simultaneous Mutithreading: A Platform for Next-Generation Processor," *IEEE Micro*, Vol. 17, No. 5, PP. 12-19, 1997.

[7] J. Singh, W. Weber, and A. Gupta, "SPLASH: Stanford Parallel Applications for Shared Memory," *Computer Architecture News*, Vol. 20, No. 1, pp. 5-44, 1992.