Raptor의 정수처리기 설계

송윤섭, 김도형, 이상원, 오형철[1], 김수원, 최성훈[2], 한우종[2]
고려대학교 전자공학과 ASIC 연구실
[1]고려대학교 정보공학과 병렬연산연구실
[2]한국전자통신연구원 컴퓨터시스템연구부
Tel. 02-923-2081, Fax. 02-928-1216, Email. rei@asic.korea.ac.kr

Design of the Integer Processor Unit for RAPTOR

Yun-Seob Song, Do-Hyeong Kim, Sang-Won Lee, [1]Hyeong-Cheol Oh,
Soo-Won Kim, [2]Seong-Hun Choi and [2]Woo-Jong Hahn
ASIC Laboratory, Department of Electronic Engineering, Korea University
[1]Parallel Computation Lab., Department of Information Engineering, Korea University
[2]Computer System Research Department, ETRI
Tel. 02-923-2081, Fax. 02-928-1216, Email. rei@asic.korea.ac.kr

ABSTRACT

This paper describes the microarchitecture of the integer processor unit of RAPTOR which is an on-chip multiprocessor.

The integer processor unit implements the 64-bit SPARC-V9 architecture and supports by hardware out-of-order instruction execution. The unit is designed to be handy so that multiple copies of the unit can be integrated with cache memories into a single chip.

The design was proceeded in a top-down manner. The hardware description and its verification were performed using Verilog-HDL.

1. 서론

RAPTOR는 비교적 단순한 구조의 스칼라 프로세서들을 하나의 칩 안에서 효과적으로 결합함으로써 슈퍼스칼라 프로세서가 가지는 구현의 어려움을 해결하고 성능 향상을 얻고자 하는 목적으로 한국전자통신연구원의 주관하에 개발되고 있는 단일 칩 다중프로세서이다.

RAPTOR의 구조는 4개의 독립된 프로세서 유닛과 1개의 그래픽 연산 유닛으로 되어 있다. 각각의 프로세서 유닛은 SPARC V9 명령어 셋 구조를 가지며 64 비트 정수 및 부동 소수점 연산을 수행한다. 또한 프로세서 유닛들은 그래픽 연산 유닛을 공유하여 그래픽 명령어를 처리한다.

본 논문은 RAPTOR의 프로세서 유닛을 이루는 정수 처리기의 설계와 검증에 관하여 논한다. 본 논문에서 다루는 정수처리기는 SPARC V9 명령어 셋을 구현한 64 비트 프로세서로서 RAPTOR의 구조에 적합하도록 단일 파이프라인으로 구성되며 비 순차적 명령어 종료를 처리하는 구조적 특징을 가진다.

2. 정수처리기의 구조

RAPTOR의 구조는 4개의 프로세서 유닛이 하나의 그래픽 연산 유닛을 공유하는 구조로서 명령어의 비순차적 종료를 하드웨어적으로 처리하지 않을 경우 많은 성능의 저하가 발생할 수 있는 구조이다. 또한 4개의 프로세서 유닛과 그래픽 연산 유닛, 외부 캐쉬 제어 유닛 등의 많은 기능 블록들을 포함하기 때문에 구현된 논리 회로의 실제 크기가 매우 중요한 요소이다. RAPTOR의 정수처리기는 SPARC 아키텍쳐 V9의 명령어 셋을 사용하며 64 비트 연산을 수행한다.

이상의 RAPTOR의 구조에서 요구되는 정수처리기의 사양을 정리하면 다음과 같다.

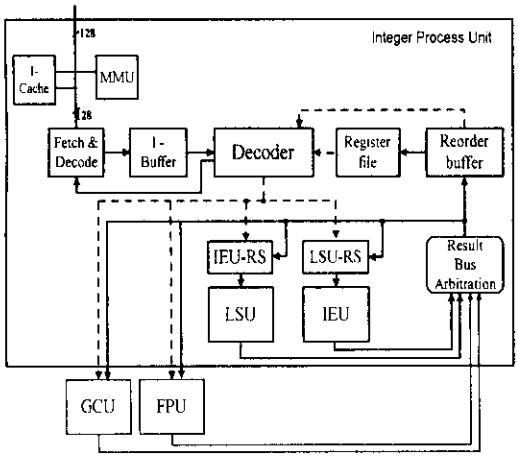
- 명령어의 비순차적 종료를 하드웨어적으로 처리한다.
- 구현된 회로의 크기가 작아야 한다.
- SPARC V9 명령어 셋을 가진다.

2.1 정수처리기의 블록도

[그림 1]은 설계된 정수 처리기의 블록도이다. 명령어는 명령어 캐쉬로부터 순차적으로 패치되어 디코더에서 해석된다. 디코더에서는 명령어 해석과 동시에 연산자를 레지스터 파일에서 읽어와 해당 연산 유닛의 레저베이션 스테이션(Reservation Station)으로 이슈한다. 레저베이션 스테이션은 명령어의 비순차적인 완료와 데이터의 의존성으로 인한 디코딩의 정지를 최소화하여 명령어 파이프 라인의 성능을 높이는 기능을 한다.

명령어는 레저베이션 스테이션으로부터 해당 연산 유닛으로 디스패치 된다. 정수처리기의 연산 유닛은 정수 실행 유닛(Integer Execution Unit : IEU), 로드 스토어 유닛(Load Store Unit : LSU)으로서, 연산 유닛에서 실행된 명령어의 결과는 리오더 버퍼(Reorder Buffer)에 쓰인다. 리오더 버퍼는 비순차적으로 완료된 연산 결과를 순차적으로 레지스터 파일에 저장하고 분기 예측이 틀렸거나 트랩이 발생하는 경우에 생기는 문제를 해결하는 기능을 한다.

면적과 성능향상의 효율성을 고려해서 리오더 버퍼는 한 개의 쓰기 포트를 갖도록 설계되었다. 그러나 결과 버스의 수 보다 연산 유닛의 수가 많으므로 포트의 사용 요구 충돌이 발생한다. 따라서 각 연산 유닛의 결과 버스 사용 요구를 중재하기 위해 결과 버스 중재 유닛(Result Bus Arbitration Unit)을 두었다. 각 연산 유닛의 명령어 수행 결과는 결과 버스 중재 유닛의 중재를 통하여 리오더 버퍼에 저장되고, 리오더 버퍼로부터 순차적으로 레지스터 파일에 쓰여지게 된다.



[그림 1] RAPTOR의 정수처리기의 블록도

2.2 파이프라인 스테이지

정수 처리기의 각 연산 유닛은 서로 다른 파이프라인 구조를 가진다. 정수 실행 유닛은 [그림 2-a]와 같이 5단계의 파이프라인을 가지며, 로드 스토어 유닛은 [그림 2-b]와 같이 7단계의 파이프라인을 갖는다. 그림에서 F, PD, D는 각각 패치, 프리디코드, 디코드를 의미하며, 명령어 캐쉬로부터 명령어를 가져온 후 각각의 기능 유닛에서 실행이 가능하도록 명령어의 디코딩을 수행하는 전 처리 단계들이다. E, MA, MD는 실행, MMU(Memory Management Unit)액세스, 데이터 캐쉬 액세스를 의미하며 명령어에 대한 실제 연산을 수행하는 단계이다. W단계는 연산 결과의 쓰기 단계로서 연산 유닛에서 연산된 결과 값이 리오더 버퍼에 저장되는 단계이다. 만약 명령어의 수행과정에서 트랩이 발생하는 경우에는 W단계에서 리오더 버퍼에 이를 알린다.

[그림 2-a] 정수 실행 유닛의 파이프라인 단계

[그림 2-b] 로드 스토어 유닛의 파이프라인 단계

2.3 연산 유닛

정수 처리기를 구성하는 연산 유닛은 정수 실행 유닛과 로드 스토어 유닛이다. 일반적으로 로드 스토어 명령어는 처리 명령어의 개수가 많으므로, 로드 스토어 유닛을 따로 분리함으로써 처리 효율을 높이고 메모리 접근 시간의 지연에 의한 성능 저하를 줄이도록 하였다.

2.3.1 정수 실행 유닛

정수 실행 유닛은 프로세서 내에서 정수 산술 연산과 논리 연산, 쉬프트 연산 등을 처리하는 곳이다. 전체 구조는 덧셈과 뺄셈을 계산하는 블록과 논리 연산을 하는 블록과 쉬프트 연산을 하는 블록이 나누어져 있다.

곱셈과 나눗셈 연산은 전용의 처리 블록을 따로 두지 않고, 기존의 덧셈과 뺄셈 계산 블록과 이것을 제어하는 블록을 사용해서 처리하도록 설계함으로써, 실제 하드웨어를 줄여서 작은 크기의 회로를 요구하는 설계 사양에 적합하도록 하였다.

2.3.2 로드 스토어 유닛

로드 스토어 유닛은 로드, 스토어 명령을 수행하여 메모리와 레지스터 간의 데이터 전송을 처리하는 기능을 한다. 로드 스토어 유닛의 파이프 라인에서는 명령어의 처리가 3단계로 이루어진다. 처음의 E(Execution) 단계에서는 로드 또는 스토어 할 메모리 주소의 계산이 이루어지고, MA 단계에서는 가상 주소가 MMU를 거쳐 실제 주소로 변환되며, MD 단계에서 데이터 캐쉬의 액세스가 일어난다.

명령어 패치, 데이터 읽기/쓰기의 액세스 충돌과 중재 지연을 피하기 위해서 데이터 캐쉬와 명령어 캐쉬는 분리하여 설계하였다. 또한 외부 캐쉬 접근 시간을 감안할 때, 스토어 미스가 발생시 다음 명령어의 지연을 방지하여 전송 효율과 파이프라인 수행 효율을 높이기 위해서 데이터 캐쉬 모듈에 여러 종류의 쓰기 버퍼(Write Buffer)를 두었다.

2.4 비순차적 명령어 수행 처리

본 정수처리기에서는 성능 향상을 위해서 명령어의 비순차적인 수행 완료가 가능하도록 하였다. 이를 위해서 별도의 하드웨어 장치인 리오더 버퍼와 레저베이션 스테이션을 두었다.

2.4.1 리오더 버퍼

명령어의 수행 완료가 순차적으로 이루어지지 않을 경우, 수행 결과에 대한 레지스터 파일의 갱신과 상태 변화는 디코딩 된 순서로 이루어지지 않는다. 따라서 예측이 틀린 분기명령어가 디코딩 되거나 트랩이 발생하는 경우에는 잘못된 레지스터 파일의 내용을 가지고 이들을 처리해야 하는 경우가 생기게 된다.

이와 같은 경우에 순차적인 수행 순서대로 레지스터 파일의 값을 갱신하며, 트랩 처리와 분기시 생기는 문제를 해결하기 위하여 본 연구의 정수 처리기에서는 리오더 버퍼를 사용하였다[1][7].

리오더 버퍼는 각 연산 유닛과 레지스터 파일 사이에 위치하며, 디코딩 되는 명령어의 순서대로 리오더 버퍼 엔트리를 할당하며 비순차적으로 완료되는 결과 값들을 버퍼링하므로써 레지스터파일이 갱신되는 순서를 순차적으로 바꾸어 주는 역할을 한다.

리오더 버퍼의 엔트리 수는 디코더 엔트리 수와 명령어 이슈 정책에 의해 결정된다. 연구에 따르면 순차적인 명령어 디스패치의 경우 리오더 버퍼 엔트리 수를 어느 정도 이상으로 올려주게 되면 더 이상 성능 향상이 이루어지지 않음을 알 수 있다[1]. 또한 유닛의 면적을 고려해서, 본 연구의 리오더 버퍼는 8개의 엔트리를 갖도록 설계하였다.

리오더 버퍼를 사용함으로써 트랩 발생시, 프리사이즈(Precise)한 트랩 처리를 하도록 하였다. 또한 분기 예측이 잘못된 분기 명령어의 경우 그 명령어의 리오더 버퍼 엔트리보다 나중에 할당된 명령어들의 수행을 무효화 시켜서 잘못 디스패치된 명령어의 진행을 복구시킬수 있다.

2.4.2 레저베이션 스테이션

데이터의 의존성으로 인한 디코더의 정지는 성능 저하의 큰 요인이 될 수 있다. 정수처리기의 각 연산 유닛은 파이프라인 깊이와 수행 사이클 수가 서로 다르므로, 명령어 피연산자와 결과 값과의 상호 의존성이 존재할 경우 명령어의 디스패치율이 크게 저하된다. 본 정수 처리기에서는 레저베이션 스테이션을 사용해서 이를 개선하였다.

레저베이션 스테이션의 각 엔트리는 각 연산 유닛으로 이슈 할 명령어들의 피연산자와 수행에 필요한 여러 정보를 저장하며, 피연산자는 레저베이션 스테이션에서 데이터의 의존성이 해결될 때까지 기다릴 수 있다. 따라서 데이터 의존성으로 인한 디코더의 정지를 최소화 할 수 있다. 레저베이션 스테이션의 엔트리 수는 각 연산 유닛에서 처리될 SPARC V9 명령어들의 분포와 종류를 고려해서 정수 실행 유닛과 로드 스토어 유닛에 각각 4개를 두었다[5].

명령어 데이터의 의존성을 빠른 시간 내에 해소하기 위하여 포워딩(Forwarding)기법을 사용하였다. 이는 의존성이 있는 피연산자의 경우 레지스터에 결과 값이 쓰여지기를 기다리는 대신, 연산이 완료되어 리오더 버퍼로 전송됨과 동시에 레저베이션 스테이션으로 결과 값을 보내어 사용함으로써 수행 시간을 단축시키도록 하였다.

2.5 결과 중재 방법

정수처리기는 하나의 디코더를 가지므로 연산 결과 값은 최대 매 클럭 당 하나가 되나, 연산 유닛간의 처리 속도의 차이로 인하여 두 개 이상의 연산 유닛이 동시에 결과 데이터 쓰기를 요구할 수 있다. 이러한 요구를 만족시켜 처리 속도를 최대로 하려면, 연산 유닛의 개수만큼 결과 버스의 쓰기 포트를 두어야 하지만, 하나 이상의 결과 버스를 가지는 방법에 의한 처리 속도의 향상은 크게 기대할 수 없으며 그에 비해

요구되는 하드웨어는 매우 커지게 됨을 이전의 연구에서 예상할 수 있다[1].

따라서 본 연구에서는 하나의 결과 버스를 두고, 정수 실행 유닛, 로드 스토어 유닛 등의 실행 유닛의 연산 결과를 레저베이션 스테이션으로 포워딩하고 리오더 버퍼로 쓰기 위한 데이터 패스(Data Path)의 사용권을 중재(Arbitration)하기 위해서 결과 버스 중재 유닛(Result Bus Arbitration Unit)을 두었다.

3. 설계 및 검증

본 정수처리기의 설계는 탑-다운(Top-Down) 방식으로 진행되었다. 먼저 주어진 설계사양에 따라 전체 블록을 기능별로 세분화하여 기능별 유닛들과 이들 유닛간의 인터페이스 신호를 정의한 후에, 정의된 유닛의 기능을 Verilog HDL을 사용해서 기술하였다.

먼저 각각의 유닛별로 주어진 입력에 따라 예측한 결과값이 나오는지를 살펴서 기능 검증을 하였다. 각 유닛의 기능 검증이 끝난 후에는 통합하여 전체 블록을 검증하였다. 이렇게 전체 설계를 세부 유닛으로 모듈화하여 진행함으로써 설계 오류의 진단과 수정, 기능의 향상 및 추가 등이 용이하도록 하였다.

통합 검증은 특정 명령어의 처리 기능과 처리 결과의 확인에 앞서, 설계된 데이터 패스가 올바로 동작하는지를 테스트하여, 유닛간의 인터페이스를 확인하고 레지스터 파일의 순차적 갱신 기능을 검증하였다. 그 후에 SPARC V9의 명령어 중 정수 명령어를 차례로 수행하여 명령어 각각의 기능 검증을 수행하였고, 기능 검증이 끝나면 데이터 의존성 문제 처리 등 명령어의 조합으로 테스트가 가능한 항목들을 검증하였다.

4. 결론 및 향후 계획

본 연구에서는 단일 칩 다중프로세서인 RAPTOR에 적합한 구조의 정수 처리기를 설계하였다. 설계된 정수 처리기는 SPARC 아키텍처 V9의 명령어 셋을 구현한 64비트 프로세서로서 리오더 버퍼와 레저베이션 스테이션을 두어서 명령어의 비 순차적인 완료가 가능하도록 하드웨어적으로 구현하였다.

향후 설계한 정수 처리기를 논리 합성 프로그램을 사용하여 실제 하드웨어로 합성하는 것을 진행중이며, 성능 향상을 위해 다중 쓰레드 처리가 가능하도록 구조를 개선하는 방법을 계획중이다.

5. 참고문헌

1. Mike Johnson, *Superscalar Microprocessor Design*, Prentice Hall, 1991.

2. David A. Patterson, and John L. Hennessy, *Computer Architecture A Quantitative Approach*, 2nd Ed, Morgan Kaufmann Publishers, Inc, 1996.

3. Saman P. Amarshinghe et al., "Multiprocessors From a Software Perspective," IEEE Micro, Vol.8, No.6, pp.52-61, June 1996.

4. L. Gwennap, "Microprocessor head toward MP on a chip", Microprocessor report, Vol.8, No.6, pp.18-21, May 1994

5. M. Tremblay, "ULTRASPRAC I : A Four-Issue Supporting Multimedia", IEEE Micro, Vol.16, No.2, pp.42-50, April 1996.

6. David L.Weaver, Tom Germond, *The SPARC Architecture Manual Ver.9*, SPARC International Inc., Prentice Hall, 1994

7. James E. Smith, Andrew R. Pleszkun, Implementing Precise Interrupts in Popelined Processors, IEEE Tran., Vol.37, No.5, pp.562-573, May 1998

8. 다중프로세서 구조 연구 최종연구보고서, 한국전자통신연구원, 1997

9. 이상원, 김영우, 오형철, 김수원, 한우종, 윤석한, "단일 칩 다중프로세서의 설계", 1998년도 대한전자공학회 추계학술대회, 1998. 11.

10. Kai Hwang, *Computer Arithmetic-Principles, Architecture AND Design*, John Wiley & Sons, 1979