SMT 마이크로프로세서에 적합한 Non-Blocking 명령어/데이터 캐쉬 제어기의 설계

이승필, 문병인, 김문경, 홍인표, 이용석
연세대학교 전기전자공학과
02-2123-2872, 02-312-4584, madoka87@dubiki.yonsei.ac.kr

Design of a Non-Blocking Instruction and Data Cache Controller for SMT Microprocessors

Seung-Pil Lee, Byung-In Moon, Moon-Gyung Kim, In-Pyo Hong, Yong-Surk Lee
Dept. of Electrical and Electronic Eng., Yonsei University
02-2123-2872, 02-312-4584, madoka87@dubiki.yonsei.ac.kr

Abstract

In this paper, a cache controller suitable for Simultaneous Multi-Threading (SMT) microprocessors is presented. The purpose of SMT architecture is to improve the performance of superscalar microprocessors by executing multiple threads at the same time. This cache controller adopts non-blocking method to process other cache requests in case of miss and multi-banking method to manage numerous cache requests effectively. Thus, because the cache controller can manage cache requests under miss, the performance of SMT microprocessors with many cache requests can be improved.

1. 서론

컴퓨터의 핵심 부품인 마이크로프로세서는 집적기술의 발달과 새로운 여러 가지 구조를 도입하면서 더욱 빠른 처리 속도를 가지게 되었다.[1] 고성능의 마이크로프로세서들은 성능의 향상을 위해 명령어 단위 병렬성(ILP)을 이용한 슈퍼스칼라 구조를 채택하고 있는데, 이는 그 자체에 한계점을 지니고 있다. 이러한 한계점을 뛰어 넘을 수 있는 스레드 단위 병렬성(TLP)을 이용하기 위한 여러 가지 방안들이 제시되어 왔다. 그 중에 하나로 Simultaneous Multi-Threading(SMT) 구조가 제안되었는데, 이러한 SMT 구조는 ILP 와 TLP 의 특성을 최대한으로 활용함으로써 그 성능의 향상을 꾀하고 있다.[2]

하지만 이러한 마이크로프로세서의 발전에도 불구하고, 메모리의 발전속도는 이에 크게 못 미치고 있다. 그 속도상의 격차로 인해 마이크로 프로세서로부터의 명령어와 데이터의 요청은 마이크로프로세서의 대기시간을 수반하게 되고, 이로 인해 전체 컴퓨터 시스템의 성능저하를 초래하게 되었다. 이러한 속도상의 격차를 줄이기 위해 캐쉬(cache) 메모리를 도입하게 되었다. 이 계층적으로 이루어진 메모리 시스템은 마이크로프로세서의 효율은 최대한 높여주면서 적은 비용으로 구현 가능하게 하였다. 하지만 캐쉬 메모리 내에 얻고자 하는 데이터가 없다면, 이 역시 캐쉬 미스를 처리 하는 동안 마이크로프로세서는 정지하고 있어야 하고, 이로 인해 성능 저하가 일어나게 된다.[1]

본 논문에서는 SMT 마이크로프로세서에 적합한 non-blocking 캐쉬 제어기를 설계한다. 이 캐쉬는 MSHR 이라는 특수한 레지스터가 존재하여 미스 처리 중에도 연속해서 캐쉬 요청을 처리 할 수 있으며, 멀티 뱅크(multi-bank)를 도입하여 뱅크 충돌이 없을 때는 최대 두개의 캐쉬 요청을 처리 가능하다.

본 논문은 총 5 장으로 구성되어 있다. 2 장에서는 SMT 마이크로프로세서의 개요에 관해 다루고, 3 장에서는 실제 캐쉬의 구조에 대해 다룬다. 4 장에서는 시뮬레이션 및 합성결과를 보인 후, 마지막 5 장에서 결론을 맺는다.

2. SMT 마이크로프로세서

오늘날, 고성능 마이크로프로세서들이 대부분 성능 향상을 목표로 채택하고 있는 구조는 슈퍼스칼라구조이다. 이것은 ILP 를 이용한 것이지만, 그 자체로 한계점을 가지는데 명령어 이슈 대역폭을 넓힌다 해도 IPC(Instruction Per Cycle)는 2 를 넘기기 힘들고, 대역폭을 보다 넓혀도 그 성능의 향상정도는 미미한 수준이라는 것이다.[2] 이러한 시스템의 성능 향상을 위해서, 스레드 단위 병렬성(Tread Level Parallelism)을 이용하기 위한 여러 가지 방안들이 제시되고 있다. 멀티 프로세서의 경우가 그 대표적이라 할 수 있으나, 이 경우 자원의 공유가 힘들기 때문에 스레드가 한 개만 존재 할 경우 동작 유닛의 활용도가 크게 저하되게 된다.

상기된 문제점을 해결하기 위해 제시된 것이 SMT(Simultaneous Multi-Threading) 구

조이다.[2] 이 SMT 구조는 ILP 와 TLP 의 특성을 최대한 활용함으로 성능의 향상을 목표로 하고 있다. 하지만 현재까지 제안된 SMT 구조를 구현하기 위해선 하드웨어 비용이 적지 않을 것으로 예상되어지고 있다. 그래서 본 연구실에서는 최소한의 하드웨어 비용으로 SMT 구조를 구현하고자 in-order completion 형태의 프로세서 구조를 연구하고, 이를 이용한 네트워크 마이크로프로세서로의 전환도 연구하고 있다. 연구되어진 마이크로프로세서는 ARM ISA 구조를 기반으로 하고 있는 슈퍼스칼라 구조이며 SMT 구조에 알맞게 약간의 변형이 이루어져 있다. 최대 4 개의 스레드를 지원할 경우의 프로세서는 다음 그림 2-1 과 같다.

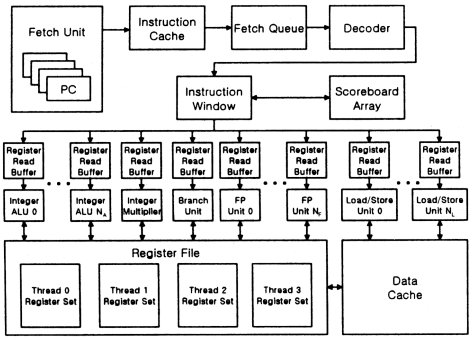

그림 2-1 4 개의 쓰레드를 지원하는 프로세서의 개략적인 구조

그림 2-1 에서처럼 명령어 캐쉬 읽기 포트. 페치 큐, 디코더 유닛, 레지스터 파일의 읽기/쓰기 포트, 명령어 윈도우, 동작 유닛, 데이터 캐쉬, 결과 버스 등의 자원은 스레드 간에 동적으로 공유되도록 하였다.

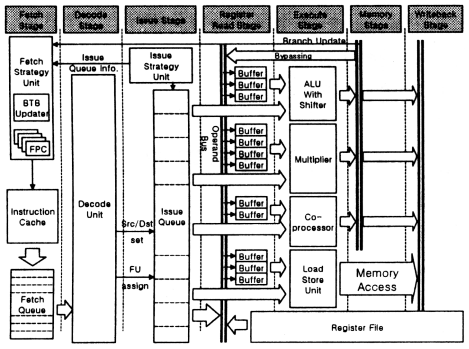

그림 2-2 파이프라인 단계 구조도

연구된 SMT 마이크로프로세서의 파이프라인 구조를 도식적으로 표현하면 그림 2-2 와 같으며 각각의 상태를 개략적으로 설명하면 다음과 같다.

- 페치 단계 (F) : 다중 스레드들로부터 명령어들을 페치 하여 페치 큐에 저장한다.
- 디코드 단계 (D) : 명령어들이 해석된다.
- 이슈 단계 (I) : 종속 문제 또는 자원 충돌 문제가 없는 명령어들이 동작 유닛들로 보내어 진다.
- 읽기 단계 (R) : 명령어 연산자들을 레지스터 파일에서 읽어 오며, 다중 사이클 명령의 경우에는 상태 머신을 통해 동작 유닛의 제어를 시작한다.
- 실행 단계 (E) : 동작 유닛에서 명령어들을 실행하고, 메모리 주소등을 계산하며, 그 결과는 결과 버퍼에 기록된다.
- 메모리 단계 (M) : 해당 메모리 주소값을 통해 데이터를 읽거나 쓴다.
- 기록 단계 (W) : 실행 결과가 순서대로 레지스터 파일에 기록된다.

3. Non-Blocking 캐쉬

3.1 Non-Blocking 캐쉬의 개요

Non-Blocking 캐쉬는 Lockup-free 캐쉬라고도 하는데, 이것은 캐쉬 미스가 발생해도 마이크로프로세서를 정지시키지 않고 MSHR(Miss information/ Status Holding Register)에 그 미스가 발생한 명령어의 정보(이를테면 캐쉬 버퍼의 주소, 입력 요청 주소, CPU 지시기, 유효 비트 등)를 저장하고 있다가 미스 순서에 의해 미스처리를 하며, 그 동안 계속 마이크로프로세서의 다음 요청을 처리한다. 캐쉬 미스 처리를 마친 후 돌아온 데이터는 이 MSHR 에 의해 CPU 로 보내어 지거나 캐쉬 메모리에 쓰여진다.[3][4]

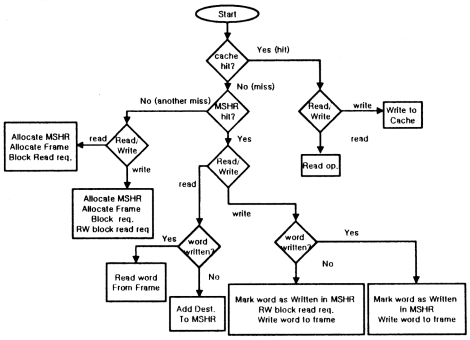

그림 3-1 캐쉬 미스 처리 흐름도

이 캐쉬 구조에서는 두 번째 미스 처리를 고려해야 한다. 이 두 번째 미스라는 것은, 한 주소의 데이터를 요청했을 때 미스가 발생하여 MSHR 에 그 정보를 저장해 놓고 그 미스처리를 하거나 미스처리를 기다리는 동안 다시 그 주소를 요청해서 또 미스가 난 경우를 말한다. 이에 따른 흐름도가 그림 3-1 에 나와있다.

3.2 2-Way Set-Associative 캐쉬의 설계

본 캐쉬 제어기는 2-way set- associative 캐쉬를 기본으로 사용하고 있다. 이 캐쉬는 direct-mapped 캐쉬 보다 캐쉬 히트율이 높다는 장점을 가진다.[1] Replacement 정책은 LRU 방식을 사용하였고, 쓰기 정책은 Write-back 방식을 적용하였다. 입력 주소는 32 비트를 받는다. 그림 3-2 는 주

소 의 구성을 보여준다.

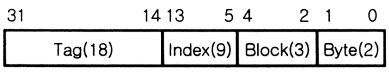

그림 3-2 캐쉬의 입력 주소 구성

명령어 캐쉬와 데이터 캐쉬, 각각 32KByte 이고, 한 라인은 32Byte 이다. 따라서 각 way 당 캐쉬 메모리의 엔트리는 2^9, 즉 512 개이다. 우선 CPU 의 IU(Instruction Unit)으로부터 주소를 입력받으면, 그 주소의 인덱스 필드로 캐쉬의 태그램을 엑세스 하여 태그램에 저장되어 있는 태그를 얻어낸다. 입력 주소의 태그와 태그램의 태그를 비교하여 태그가 같고 그 태그가 유효하다면 히트/미스 판별유닛에서 캐쉬 히트가 발생했으며 어느 쪽의 way 가 선택 되었는지 알려 준다. 그와 동시에 데이터 선택 유닛은 데이터램의 데이터를 입력받은 후 블록 주소로 올바른 데이터를 선택해서 CPU 로 보내게 된다. 만일 태그램의 태그와 입력 주소의 태그가 같지 않다면, 이것은 캐쉬 미스가 발생한 것이고, 요청된 데이터가 캐쉬 메모리에 없다는 것을 의미한다. 이때는 MSHR 로 정보를 보내고 MSHR 에서는 캐쉬 미스 처리를 해 준다. 그림 3-3 은 2-way set-associative 캐쉬의 블록도이다.

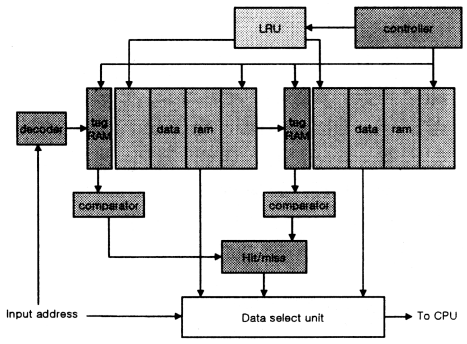

그림 3-3 2-way set-associative 캐쉬의 블록도

3.2.1 캐쉬 태그램과 데이터램

Multi-bank 를 갖는 캐쉬 메모리가 얻는 장점은 입력주소가 뱅크 충돌이 없다면 다수의 캐쉬 요청을 처리 가능하다는 것이다. SMT 마이크로프로세서의 경우 다수의 스레드가 존재하므로 그에 따른 캐쉬 요청도 잦아지게 되고, 그 캐쉬는 필연적으로 multi-port 나 multi-bank 방식을 써야 한다. 하지만 multi-port 를 갖는 캐쉬 메모리는 설계도 어렵고, 또한 가격도 상대적으로 비싸지므로, 본 캐쉬 제어기는 기존의 메모리를 이용할 수 있는 multi-bank 방식을 사용하였다.

설계된 캐쉬 제어기는 각 way 마다 2 개의 뱅크를 갖는다. 각 뱅크는 입력주소의 인덱스 필드의 최하위 주소로 나뉘어 지게 된다. 즉, 그 최하위 주소(LSB)가 0 이면 뱅크 0 에, 1 이면 뱅크 1 에 저장되고 읽혀지게 된다. 본 캐쉬는 태그램으로 18X256 램 셀이 way 당 2 개로 총 4 개가 쓰였고, 데이터램의 경우 명령어캐쉬는 way 당 64X256 램 셀이 8 개, 데이터캐쉬는 8X256 램 셀이 128 개 쓰였다. 그림 3-4 는 2 개의 뱅크를 가지는 메모리 구조를 보여준다.

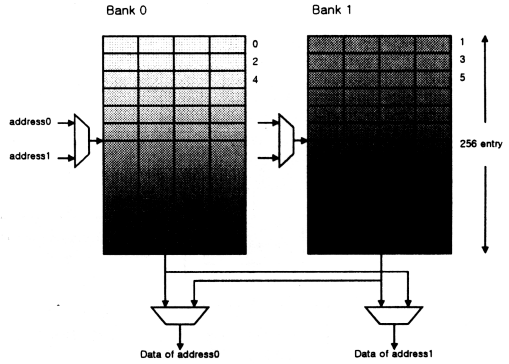

그림 3-4 2-bank 를 가지는 캐쉬메모리구조

각 뱅크는 주소의 선택을 위한 멀티플렉서와 출력 데이터를 올바른 주소로 내 보내기 위한 데이터 멀티플렉서를 가지고 있다. 이 멀티플렉서의 선택 신호는 BSU(Bank Select Unit)에서 출력하며, 또한 이 유닛에서 뱅크 충돌도 감시하여 충돌이 발생했을 때 BC(bank conflict) 신호를 출력한다. 본 캐쉬는 2 개의 캐쉬요청을 동시에 처리하기위해 비교기, 히트/미스판별유닛, DSU 가 각각 두개씩 존재한다.

3.2.2 MSHR

MSHR 은 앞서 설명한 바와 같이 미스가 발생했을 때 그 정보를 저장해 놓는 일종의 레지스터이다. 명령어 캐쉬와 데이터캐쉬의 내부 구조는 조금 차이가 있다. 그림 3-5 를 보면 위쪽의 명령어 캐쉬의 MSHR 은 유효비트와 미스발생 주소로 구성 되어있는 반면, 아래쪽 데이터 캐쉬인 경우 유효비트, 쓰기 모드 비트, 데이터 타입, 미스 주소, 데이터 또는 로드버퍼 ID 를 저장하도록 되어있다.

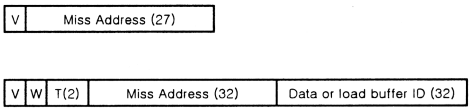

그림 3-5 명령어 캐쉬와 데이터 캐쉬의 MSHR 구조

본 논문에서 설계된 캐쉬 제어기는 미스가 발생했을 때 그 미스의 정보를 저장하는 MSHR 이 8 개를 가지고 있다. 이는 단일 스레드를 가지는 마이크로프로세서의 경우 충분한 숫자지만 본 연구실에서 연구되어지고 있는 SMT 마이크로프로세서의 경우에는 다수의 스레드가 존재하고, 그에 따라 멀티 프로세서를 가진 시스템처럼 동작하게 되므로 다소 부족할 수도

있다. 따라서 차후에 SMT 마이크로 프로세서에 적용될 캐쉬 제어기는 그 스레드 숫자에 따라 더 많은 개수의 MSHR 을 가져야 할 것이다.

MSHR 제어기는 이 8 개의 MSHR 에 미스 정보를 저장하기도 하고 미스 순서에 따라 메인 메모리로 엑세스를 하는 등 모든 MSHR 에 관련된 동작을 관리한다. 그림 3-6 은 MSHR 부분의 블록도이다.

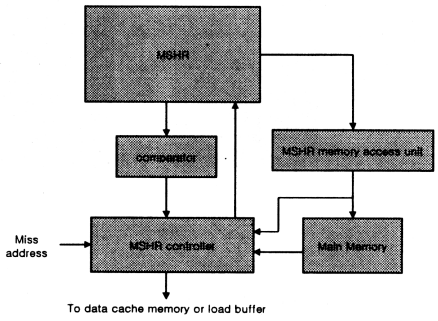

그림 3-6 MSHR 부분의 블록도

4. 시뮬레이션 및 합성결과

4.1 시뮬레이션

본 논문에서는 non-blocking 캐쉬 제어기의 동작을 확인하기 위해서 Verilog HDL 을 사용하여 기능 수준과 게이트 수준에서 시뮬레이션을 수행하여 검증하였다.

우선 인공적인 방법을 사용하여 non-blocking 캐쉬 제어기를 기술한 HDL 의 문법오류 및 기능 검증을 수행하였다.

게이트 수준에서의 시뮬레이션을 수행하기 위해 합성결과를 다시 게이트 수준의 넷 리스트로 추출한 뒤, 연결선의 지연 정보를 담고 있는 SDF (Standard Delay Format) 파일을 삼성전자의 CubicWare 의 CubicDelay 를 이용하여 만들고 두 파일을 읽어들여서 시뮬레이션 하였다. 또한 부분적으로 Altera 의 Quartus 를 이용하여 검증하였다.

4.2 합성

본 논문에서 사용된 라이브러리는 삼성전자의 0.35 ㎛공정의 std90 라이브러리를 사용하였으며 합성툴은 Synopsys 의 Design Analyzer 를 사용하였다. 메모리는 자동합성이 불가능하므로 삼성전자의 메모리 컴파일러인 std90memgen 을 사용하여 발생시켰다. 이 메모리의 지연 시간은 테스트를 통해 알아냈는데 약 6.88 ns가 걸렸다. 따라서 본 캐쉬의 최대 지연 시간을 측정할 수 있었는데 그 결과는 다음 표 4-1 과 같다.

최대 지연 경로(critical path)는 칩의 최대 동작 주파수를 결정하는 중요한 경로이다. 타이밍 시뮬레이션에 의해 이러한 경로를 찾아내는 것이 가능하다. 표 4-1 에 각 유닛의 게이트 수와 시간, 그리고 최대 지연 시간을 나타내었다. 이 결과 최대 지연 시간은 3.0V, 85℃, 최악 조건에서 9.95 ns 로, 약 100MHz 의 입력 주파수에서 동작 가능하다

표 4-1 최대 지연 경로 및 지연 시간

	게이트 수	시간	최대 지연 시간
뱅크선택 mux	22.59	0.29	0.29
캐쉬 태그램	34642.33	6.88	7.17
태그 데이터 선택 Mux	45.25	0.29	7.46
태그 비교기	74.46	1.07	8.53
히트/미스 판별 유닛	4.74	0.47	9.00
데이터 선택 유닛	954.28	0.95	9.95

5. 결론

본 논문에서는 SMT 마이크로프로세서에 적합한 캐쉬 제어기를 설계하였다. non-blocking 방식을 사용하여, 미스가 발생했을 때 미스처리 시간을 줄이고, 미스처리 중에도 다음 캐쉬 요청을 처리가 가능하고, multi-banked 방식을 사용하여 여러 개의 캐쉬 요청을 동시에 효과적으로 처리할 수 있다. 그 결과, 합성된 캐쉬 제어기는 3.0V, 85℃, 최악 공정 조건에서 9.95 ns 의 최대 지연 시간을 가지고, 이 경우 100MHz 의 입력 주파수에서 동작 가능하다. 이와 같이 본 논문에서 설계된 캐쉬 제어기는 고성능 마이크로 프로세서에 내장되어, 미스 처리 중에도 캐쉬 요청을 처리할 수 있고, 여러 개의 캐쉬 요청을 동시에 처리 가능케 함으로써, 마이크로 프로세서의 성능을 최대한으로 발휘 할 수 있을 것으로 기대된다. 또한, 본 캐쉬 제어기가 내장 될 SMT 마이크로프로세서는 차후에 패킷 스케줄러 등을 추가하여 네트워크 프로세서로 개발될 예정이다.

(Acknowledgement : 이 논문은 과학기술부의 2000 년도 국가지정연구실 사업의 지원으로 쓰여진 것입니다)

참고문헌

[1] Nikitas Alexandridis, *Design of Microprocessor Based Systems*, Prentice Hall. pp 246~312, 1993.

[2] Susan J. Eggers, Joel S. Emer, Henry M. Levy, Jack L. Lo, Rebecca L. Stamm, Dean M. Tullsen, "SIMULTANEOUS MULTI-THREADING : A Platform for Next-Generation Processors", IEEE Micro, p.p. 12~19, September/October 1997.

[3] Tien-Fu Chen, Jean-Loup Baer, "Reducing Memory Latency via Non-Blocking and Prefetching Caches", ACM SIGPLAN Notices Volume 27, Issue 9, September 1992.

[4] David Kroft, "Lockup-free Instruction Fetch/Prefetch Cache Organization", Conference proceedings of the eighth annual symposium on Computer Architecture May 1981.