ARM-SMT용 Load/Store Unit(LSU) 설계

김 재 억, 이 용 석
연세대학교 전기전자공학과
전화 : 02-2123-2872 / 핸드폰 : 016-9870-2990

Design of a Load/Store Unit for ARM-SMT Microprocessors

Jae-Eok Kim, Yong-Surk Lee
Processor Laboratory, Dept. of Electrical and Electronic Eng. Yonsei University
E-mail : bluecloud12@hanmail.net

Abstract

The superscalar architecture shows limit in performance improvement recently. While, SMT(Simultaneous Multi-Threading) architecture is receiving remark. The purpose of SMT architecture is to improve the performance of superscalar microprocessors by executing multi threads at the same time. In this paper, a load/store unit(LSU) suitable for ARM-compatible SMT microprocessors is presented. This LSU supports load instructions and store instructions of ARM ISA. This LSU keeps away the degradation of SMT by cache miss.

I. 서론

대부분의 고성능 마이크로프로세서는 슈퍼스칼라 RISC방식[1]으로 설계되고 있다. 슈퍼스칼라 방식은 ILP(Instruction Level Parallelism)를 이용하는 구조로 점차 성능 향상에 한계를 보이고 있다. 이러한 한계점을 극복하고 시스템 성능 향상을 위하여 TLP(Thread Level Parallelism)가 연구되고 있다. TLP를 사용하는 프로세서 구조 중에 SMT(Simultaneous Multi-Threading) 구조[2]가 제안되었는데, SMT 구조는 모든 스레드가 하드웨어 자원을 공유하여 ILP와 TLP를 적용함으로써 프로세서의 성능을 향상시킬 수 있다.

마이크로프로세서는 명령어 캐쉬로부터 명령어를 읽어들이고 데이터 캐쉬로부터 데이터를 읽어 들여 레지스터에 저장한 다음 이를 이용하여 동작한다. 현재 마이크로프로세서와 메모리의 속도는 10배 이상 차이가 나나 대다수의 컴퓨터시스템에서는 캐쉬를 이용하여 이 문제를 극복하고 있다. 하지만 TLB miss나 캐쉬 미스가 발생한 경우 어쩔 수 없이 데이터를 읽어들이는데 많은 시간이 소요되어 파이프라인이 스톨되며 이는 마이크로프로세서의 성능을 떨어뜨린다. ILP 및 TLP를 사용하는 SMT 구조의 경우 보다 많은 데이터를 읽어들이기 때문에 메모리 액세스 문제가 전체 프로세서 성능 향상을 제한한다. 그러므로 프로세서 성능 향상을 위해서는 데이터 액세스 문제로 인한 성능 저하를 최소화할 수 있는 LSU가 필요하다.

본 논문에서는 ARM-ISA[3] 규격을 지원하는 SMT 프로세서에 적합한 LSU를 설계한다. 이 LSU는 데이터 액세스 문제가 발생했을 시에도 내부 파이프라인 단계에서 스톨이 발생하지 않게 설계되었다. LSU에는 스테이트 머신이 내장되어 한 명령어로 다수의 데이터를 액세스하는 Block data transfer명령어를 지원한다. 전체적인 설계를 간단하기 위하여 다른 유닛과 중복되는 부분은 줄이고 다른 유닛들과 유기적으로 작동된다.

본 논문은 총 5장으로 구성되어 있다. 2장에서는 SMT 마이크로프로세서의 개요에 대하여 다루고 3장

에서는 LSU의 구조에 대하여 다룬다. 4장에서는 시뮬레이션 및 합성 결과를 보인 후, 마지막 5장에서 결론을 맺는다.

II. SMT 마이크로프로세서

오늘날 대다수의 고성능 마이크로프로세서가 사용하고 있는 구조는 슈퍼스칼라구조이다. 이는 ILP를 이용한 것으로 더 이상의 성능 향상은 어렵다. 명령어 이슈 대역폭을 늘려도 IPC(Instruction Per Cycle)은 2를 넘기기 힘들고, 대역폭을 보다 넓혀도 그 성능의 향상도는 낮다. 이러한 시스템의 성능 향상을 위해서, TLP를 이용하기 위한 여러 가지 방안들이 제시되고 있다. 멀티프로세서의 경우가 가장 대표적이나 자원의 공유가 힘들기 때문에 단수의 스레드가 존재할 경우 동작 유닛의 활용도가 저하된다.

이런 문제점을 보안한 구조가 SMT구조이다. SMT구조는 모든 자원을 공유하여 ILP와 TLP를 최대한 활용하여 성능을 향상시킨다. 하지만 SMT구조를 구현하기 위해선 하드웨어 비용이 적지 않게 필요하다. 최소한의 하드웨어 비용으로 마이크로프로세서를 구현하고자 in-order issue, in-order completion 형태의 SMT가 제안되었으며 ARM ISA 구조를 기반으로 하고 있는 슈퍼스칼라 구조를 SMT구조에 맞게 변형하여 설계되었다. 최대 8개의 스레드를 지원할 경우의 마이크로프로세서의 구조는 다음 그림 2-1과 같다.

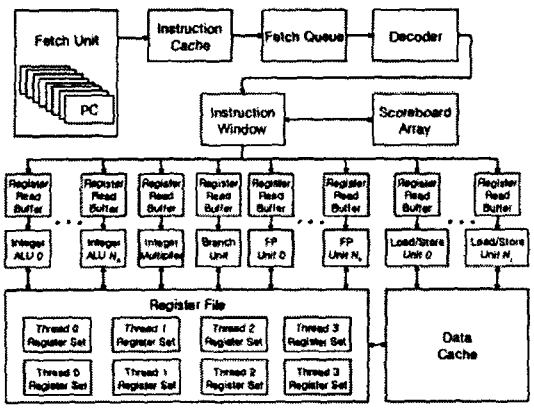

그림 2-1 8개의 스레드를 지원하는 프로세서 구조

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

SMT 마이크로프로세서는 8단계의 파이프라인 구조를 가지며 각각의 상태는 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 기능 유닛으로 보내어 진다.
- 레지스터 읽기 1단계(RR1) : 사용한 명령어 연산자들의 주소를 레지스터 파일로 전송하거나 바이패싱 정보를 사용한다. 다중 사이클 명령을 사용하는 기능유닛의 경우에는 스테이트 머신을 두어 전체 과정을 컨트롤한다.
- 레지스터 읽기 2단계(RR2) : 명령어 연산자들을 레지스터 파일에서 읽어 오거나 바이패싱 받는다.
- 실행 단계(E) : 기능 유닛에서 명령어들을 실행하고, 메모리 주소 등을 계산한다. condition code를 계산한다.
- 메모리 단계(M) : 해당 메모리 주소값을 통해 데이터를 읽거나 쓴다. 분기 예측 판별을 한다.
- 기록 단계(W) : 실행 결과가 순서대로 레지스터 파일에 기록된다.

각 파이프라인 단은 각각 스레드 ID와 PC값을 가지고 있으며 스레드 ID는 레지스터나 PC 액세스, 익셉션 발생시 같은 스레드의 명령어들을 플러쉬시킬 때 사용된다.

III. Load/store unit

LSU는 데이터 액세스에 대한 것을 관리하는 기능 유닛이다. 가장 기본적인 기능은 연산자 값으로 메모리 어드레스를 계산하여 메모리를 액세스하는 것이다. 총 5단계(RR1, RR2, E, M, W) 파이프라인을 가지고 있다. 그림 3-1은 본 LSU의 간략화 시킨 블럭도이다.

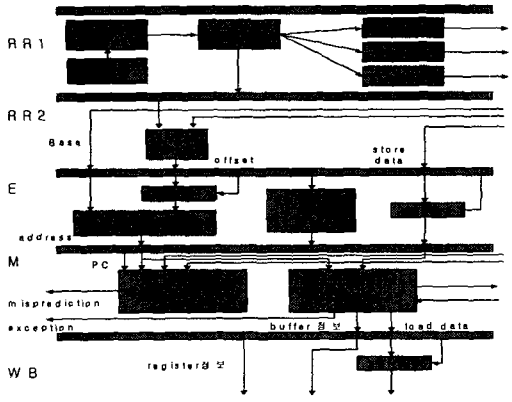

그림 3-1 LSU의 전체 블럭도

레지스터 읽기 1번 단계에서는 스테이트 머신이 내장되어 Block data transfer를 조절한다. 실행단계에는 가산기와 쉬프터가 내장되어 메모리 어드레스를 계산한다. 메모리 단계에서 캐쉬 액세스를 하며 이 때 문제가 생기면 익셉션 처리를 한다. 이외에도 ARM ISA의 특징 중 하나인 condition field의 판독기가 실행 단에 위치하며 메모리 단에 분기 예측 판단기가 위치한다.

SMT구조에서는 기능유닛의 내부 파이프라인 단계에서 스톨이 일어나는 것을 지양하고 있다. 이는 파이프라인단계에서 스톨이 발생할 경우 후속하는 파이프라인이 다른 스레드인 경우에도 영향력을 미치게 되어 TLP의 효율이 낮아지기 때문이다. 본 LSU에서는 외부 메모리 엑세스 버퍼, 로드 버퍼, 스토어 버퍼, 캐쉬 미스 주소 파일 등 4가지 외부 버퍼를 이용하여 파이프라인 내부의 스톨을 없앤다. (그림 3-2) 데이터 메모리 액세스에 문제가 발생한 경우 DTLB 미스를 제외한 경우는 LSU 파이프라인 내부에서 외부 버퍼로 정보를 보내어 처리한다. 그럼으로써 LSU 내부에 스톨의 발생을 없애고 TLP의 효율을 높였다. 그림 3-3는 데이터 메모리 엑세스하는 경우 발생하는 모든 경우를 담은 것이다.

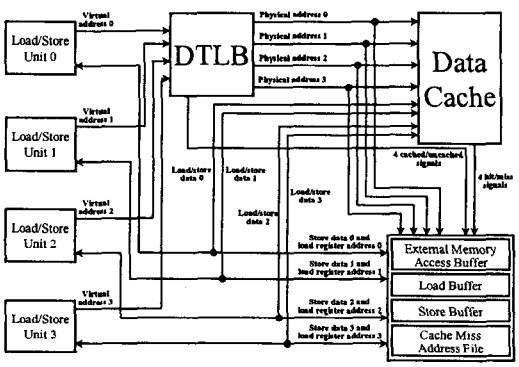

그림 3-2 LSU, 캐쉬, 외부 버퍼 연결도

LSU는 block data transfer를 구현하기 위해서 스테이트 머신을 내장하고 있다. LSU의 파이프라인은 single data transfer 기반으로 설계되어 있으며 block data transfer 명령어 수행시에는 스테이트 머신의 각

단계에 맞추어 single data transfer명령어들로써 수행되어진다. 스테이트 머신은 3bit로 이루어진 스테이트

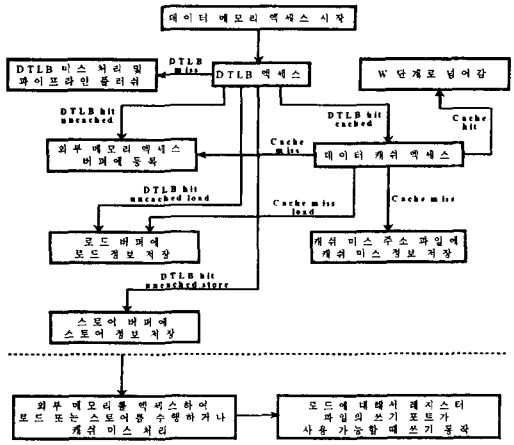

그림 3-3 데이터 메모리 액세스 과정

를 가지며 이에 따라 동작한다. block data transfer에는 Increment, Decrement, Pre-, Post-를 조합한 4가지 어드레싱 모드가 있다. 하지만 모든 어드레싱 모드를 한 가지 방법으로 구현할 있다. 최하위 레지스터가 연결될 어드레스를 베이스 어드레스로 놓고 오프셋을 계속 스테이트 머신이 끝날 때까지 증가시키는 방법을 이용하며 위의 4가지 어드레싱 모드를 간략히 구현할 있다. 로드 명령어의 경우 레지스터 리스트에 PC가 있을 경우 분기 명령이 되기 때문에 동작이 달라진다. 먼저 PC를 로드한 다음 최하위 레지스터가 연결될 어드레스를 구한 후 어드레스를 계속 증가시킨다. PC를 처음 로드하므로써 분기 예측 판별을 빠르게 수행한다. 밑의 그림 3-4와 그림 3-5는 본 LSU스테이트 머신의 스테이트 변화를 나타낸 것이다.

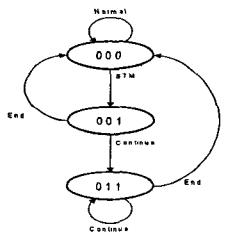

그림 3-4 STM에 관련된 스테이트 변화

LSU는 3개의 읽기 포트와 하나의 쓰기 포트를 가진다. 계산한 어드레스 값을 다시 레지스터에 저장하는 명령어들의 경우 2개의 쓰기 포트가 필요하다. 마이크로프로세서에서 포트의 증가는 하드웨어 비용을 초래한다. 그러므로 기능유닛 자체는 간단히 하고 다른 기능유닛과 유기적으로 움직여서 작동시킴으로 설계를 단순화하고 하드웨어 비용을 절감한다. 어드레스를 저

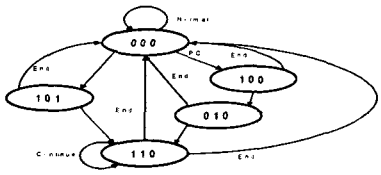

그림 3-5 LDM에 관련된 스테이트 변화

장하는 옵션이 붙은 로드 명령어의 경우 보통의 로드 명령어와 어드레스를 가감하는 명령어 2가지로 분리되어 처리한다.

메모리의 데이터를 레지스터로 옮기고 레지스터 값을 메모리에 옮기는 swap명령어의 경우에도 비슷한 원리로 동작한다. 하나의 LSU에서 처리할 경우 multi-cycle로 동작하게 되어 설계가 복잡해진다. swap의 경우에도 위와 같은 2개의 LSU를 사용하여 처리한다. 이렇게 복수의 기능유닛과 연동된 알고리즘은 이슈율을 높이고 하드웨어 설계 비용을 줄이는 데 도움이 된다. 그림 3-6, 3-7은 복수의 기능유닛을 이용하여 명령어를 처리하는 모습이다.

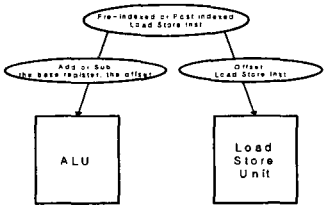

그림 3-6 Base update명령어를 ALU와 LSU로 실행

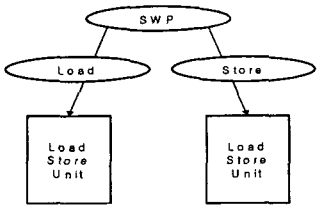

그림 3-7 SWP 명령어를 2개의 LSU로 실행

IV. 시뮬레이션 및 합성결과

본 논문에서는 LSU를 설계, 검증하는데 Verilog HDL을 사용하였다. 우선 LSU를 각 파이프라인들을 시뮬레이션으로 검증하였으며 LSU 전체에 대해서도 시뮬레이션을 수행하였다. 시뮬레이션은 functional simulation과 post-synthesis simulation을 수행하였다. 설계, 시뮬레이션을 하는데 ALDEC사의 Active HDL을 사용하였으며 합성은 Synplicity Synplify Pro 툴을 사용하였다. QuickLogic사의 QuickSD family의 FPGA device ql82sd-PB516를 선택하여 합성하였으며 이 때 파이프라인별 AREA와 최대 지연 시간은 표 4-1과 같다.

표 4-1 파이프라인별 AREA와 최대 지연 시간

파이프라인	AREA	최대 지연시간
RR1	683	19.263
RR2	354	8.757
EX	1160	36.993
MA	239	9.797
WB	104	3.039

V. 결론

본 논문에서는 ARM-ISA를 지원하는 SMT용 LSU를 설계하였다. 외부 버퍼를 두어 파이프라인 내의 스톨을 없애 TLP를 향상시켰으며 스테이트 머신을 사용하여 Block data transfer를 구현한다. 일부 명령어들의 경우는 복수의 기능유닛을 사용하여 처리함으로써 자체의 구조를 단순화하고 이슈율을 높인다. 본 논문에 설계된 LSU는 SMT마이크로프로세서에 사용되어 시스템 성능 향상에 도움이 될 것이다.

참고문헌(또는 Reference)

[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, Deam M. Tullsen, "SIMULTANEOUS MULTI-THREADING " A Platform for Next-Generation Processors", IEEE Micro, p.p. 12~19, September/October 1997.

[3] ARM, ARM Architecture Reference Manual, Part A CPU Architecture, 1996.