論文 95-5-5-12

펜티엄 마이크로프로세서 데이터 캐쉬 제어기에 관한 연구

正會員 李 溶 錫*

A Study on the Data Cache Controller of the Pentium Microprocessor

Yong Surk Lee* Regular Members

要 約

본 논문에서는 한 클럭 사이클에 두개의 데이터를 동시에 엑세스할 수 있는 펜티엄 마이크로프로세서의 데이터 캐쉬 제어기의 구조를 제안하고 이를 Verilog HDL(Hardware Description Language)로 기술하여 기능을 검증하였다. 데이터 캐쉬 제어기는 캐쉬 제어 블럭, 버퍼 블럭, MUX/정렬 블럭, 버스 제어 블럭, 캐쉬 메모리 블럭으로 구성된다. 데이터 캐쉬 제어기는 선형 주소 생성기로부터 두개의 선형 주소를 받아 필요한 데이터를 두개의 실행유닛에 제공한다. 데이터 캐쉬는 8개의 뱅크(bank)로 구성된 단일 포트 구조로 설계하였다. 필요한 데이터의 주소가 정렬되어 있고, 캐쉬 미스가 일어나지 않으면 캐쉬는 한 사이클에 두개의 데이터를 읽거나 쓴다. 읽기 미스 시에 캐쉬 제어기는 라인 버퍼 캐싱을 이용하여 캐쉬 미스 페널티를 줄여 준다. 쓰기 동작시 캐쉬는 writeback 기법으로 외부 버스 이용횟수를 줄여 주고, buffered write-through 기법을 사용하여 실제 외부 메모리로 쓰기 동작이 완료되지 전에 다른 읽기 또는 쓰기 동작을 행한다. 캐쉬 데이터의 일관성 유지를 위해 스누핑과 MESI 프로토콜을 채택하였다. 이 데이터 캐쉬 제어기는 상공자원부의 펜티엄 설계 연구 과제의 일부로 설계되었으나, 한 클럭 사이클에 두 개의 데이터 엑세스를 요구하는 일반적인 수퍼스칼라 마이크로프로세서에도 적용할 수 있다.

ABSTRACT

This paper proposes a data cache controller of the Pentium microprocessor which can access two data in a clock cycle. It is functionally described and verified with Verilog HDL. The data cache controller consists of a cache control block, a buffer block, a MUX/align block, a bus control block, and a cache memory block. It receives two linear addresses from the address generation unit and sends out the requested data to the dual-execution unit. Data cache is designed with single ported, 8 bank interleaved

*연세대학교 전자공학과
Dept. of Electronic Eng., Yonsei University
論文番號: 94288-1017
接受日字: 1994년 10월 17일

structure. If the addresses of the data are aligned and no cache miss occurs, it reads or writes two data in one cycle. During a read miss, line buffer caching is used to reduce the cache miss penalty. During a write cycle, external bus utilization is reduced by using writeback policy. Also buffered write-through technique is used to read or write before the write to external memory completes. Snooping and MESI protocol is used for maintaining the cache coherency. This data cache controller is designed for the Pentium microprocessor design project sponsored by the Ministry of Commerce and Industry. However, the research results can be applied to general superscalar microprocessors which require two data accesses in a clock cycle.

Ⅰ. 서론

현재 컴퓨터 시스템의 마이크로프로세서는 집적회로 설계 기술의 발달로 성능이 급격히 향상되고 있다. 그러나 메인 메모리의 엑세스 시간은 마이크로프로세서의 처리 시간보다 상당히 길어서 컴퓨터 시스템에 병목 현상이 일어난다. 이러한 문제는 캐쉬 메모리의 도입으로 많은 개선을 이루었다.[1] 현재 대부분의 고성능 마이크로프로세서는 캐쉬 메모리를 내장하거나 외부에 장착 또는 내부와 외부에 모두 장착한다.

마이크로프로세서의 CPI(Clock Per Instruction)를 줄이는 방향으로 1970년대 후반부터 파이프라인 기법이 사용되어 한 사이클에 하나의 명령어를 처리하였다. 그 후 1980년대 후반 부터는 여러 개의 파이프라인을 사용하여, 한 사이클에 다수의 명령어를 동시에 처리해 주는 수퍼스칼라 기법이 등장하여 마이크로프로세서의 처리 속도는 더욱 향상되었다. 수퍼스칼라 마이크로프로세서에서는 동시에 코드 읽기와 데이터 읽기가 발생한다. 따라서 마이크로프로세서 내부의 코드 캐쉬와 데이터 캐쉬를 분리하여 상호 충돌을 줄이는 하버드 아키텍처를 사용하여, 파이프라인 멈춤을 해결해 주었다.[2]

본 논문에서는 Intel의 x86 구조와 호환성을 가지면서, 수퍼스칼라 기법을 사용한 펜티엄 마이크로프로세서의 내부 데이터 캐쉬를 엑세스하는 캐쉬 제어기를 제시한다. 본 논문에서 다루는 펜티엄 마이크로프로세서는 두개의 프리펫치 버퍼, 주소 발생 회로, 실행유닛 및 데이터 캐쉬 인터페이스를 가지고 있다. 캐쉬 제어기는 주소 발생 회로로 부터 한 개 또는 두 개의 선형 주소를 받아 데이터 캐쉬 메모리를 엑세스하여 필요한 데이터를 실행유닛 또는 FPU 에 제공한다. 수퍼스칼라 데이터 캐쉬 메모리 구조로는 이중 포트 또는 interleaved 단일 포트 방식이 가능하다. 본 연구에서는 면적과 전력면에서 유리한 interleaved 단일 포트 구조를 선택하였다. 필요한 데이터의 주소가 정렬되어 있고, 뱅크 충돌(bank conflict)이 발생하지 않고, 캐쉬 미스가 일어나지 않으면 한 사이클에 두 개의 데이터를 읽거나 쓰는 것이 가능하다.[3] 캐쉬 미스가 일어나면 미스 처리 관계로 여러 사이클이 지연되나, line fill 버퍼와 쓰기 버퍼(write buffer)를 사용하여 읽기 동작에서는 3 사이클, 쓰기 동작에서는 1 사이클로 캐쉬 미스 페널티를 최소화하였다. 캐쉬를 사용할 때 생기는 캐쉬 일관성 문제는 스누핑과 MESI 프로토콜로 해결을 하였다.

Ⅱ. 수퍼스칼라 파이프라인

기존의 i486과 같은 마이크로프로세서는 단일 파이프라인 구조를 채택하여 대부분의 명령을 한 클럭사이클에 처리하였다. 한 클럭사이클당 1개 이상의 명령을 실행하기 위해서는 수퍼스칼라 아키텍처와 같은 구조로 마이크로프로세서가 설계되어야 한다. 본 논문에서 다루는 펜티엄 마이크로프로세서는 A-파이프, B-파이프의 두 개의 파이프라인을 가진다. 두 개의 파이프라인은 동시에 독립적으로 명령을 실행하여 클럭사이클당 2개의 명령을 처리한다. 펜티엄 마이크로프로세서의 블럭도는 그림 1.과 같다. 각 파이프라인은 PF(prefetch), D1(decode), D2(address generate), EX(execution), WB(writeback)의 다섯 단계로 구성이 된다.[4]

PF 단계는 명령어 캐쉬로부터 32바이트의 캐쉬 라인을 프리펫치 버퍼로 가져온다. D1단계는 버퍼로부터 명령어를 가져와 디코더에서 종속성을 판단하고 명령어를 해석한다. D2단계는 데이터 캐쉬로부터 데이터를 가져

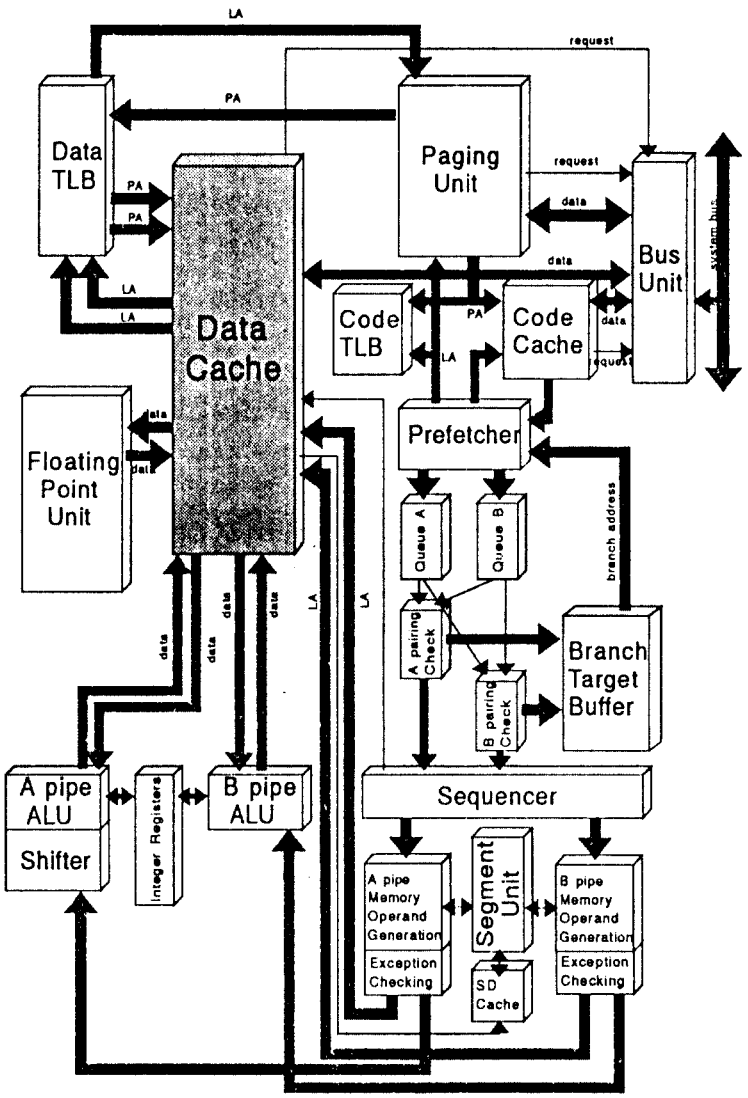

그림 1. 펜티엄 마이크로프로세서의 블럭도
Fig. 1. Block diagram of the Pentium microprocessor

오기 위한 선형 주소를 생성하고 제어 유닛에서 나온 제어신호들을 각 유닛내에서 국부해석하여 실제 제어 신호를 생성한다. EX 단계는 ALU 연산 또는 데이터 캐쉬를 엑세스한다. WB 단계는 레지스터 화일에 ALU 연산 결과를 저장하고 플래그값을 갱신하여 명령어의 수행을 종료한다. 모든 파이프라인 단계는 쌍을 이룬 두 명령어가 모두 수행이 끝날 때까지 진행하지 못하고 항상 쌍을 이루어 진행한다.

본 연구에서 다루는 데이터 캐쉬 제어기는 D2 단계에서 두 개의 선형주소를 받아 EX 단계에서 데이터 캐쉬

를 엑세스하여 데이터를 실행 유닛 및 FPU에 제공한다. 캐쉬 읽기 또는 쓰기 동작에서 캐쉬 히트가 일어나면, EX 단계의 한 사이클후 WB 단계로 파이프라인이 진행을 하고, 캐쉬 미스가 일어나면 파이프라인은 멈추게 되어, 캐쉬 미스가 처리되는 동안 EX 단계는 3 사이클이상이 걸리게 된다.

Ⅲ. 데이터 캐쉬 제어기

데이터 캐쉬 제어기는 그림 2.와 같이 캐쉬 제어 블럭, 캐쉬 메모리 블럭, 버퍼 블럭, MUX/정렬 블럭, 버스 제어 블럭으로 구성된다.

1. 캐쉬 제어 블럭

캐쉬의 모든 동작을 제어하는 부분이 캐쉬 제어 블럭이다. 시퀜서로부터 필요한 사이클의 요청을 받아서 읽기, 쓰기, 스누프 동작을 위한 캐쉬 엑세스 주소와 제어 신호들을 생성시킨다. 데이터 캐쉬 제어기가 처리하는 사이클은 다음과 같다.

.플러시 사이클
.무효화 사이클
.라인 교체 사이클
.A-파이프 읽기/쓰기
.B-파이프 읽기/쓰기

플러시 사이클은 데이터 캐쉬에 있는 데이터를 전부 무효화시키는 동작으로 프로세서의 reset시에 수행이 된다. 무효화 사이클은 데이터 캐쉬의 한 라인을 I(invalid) 상태로 만드는 작업이다. 라인 교체 사이클은 읽기 미스가 일어났을 때 라인 버퍼로 읽어온 외부 데이터를 캐쉬 메모리에 저장하는 작업으로, 캐쉬 메모리의 양쪽 way가 모두 채워져있을때, LRU(Least Recently Used) 알고리즘에 의해 가장 최근에 적은

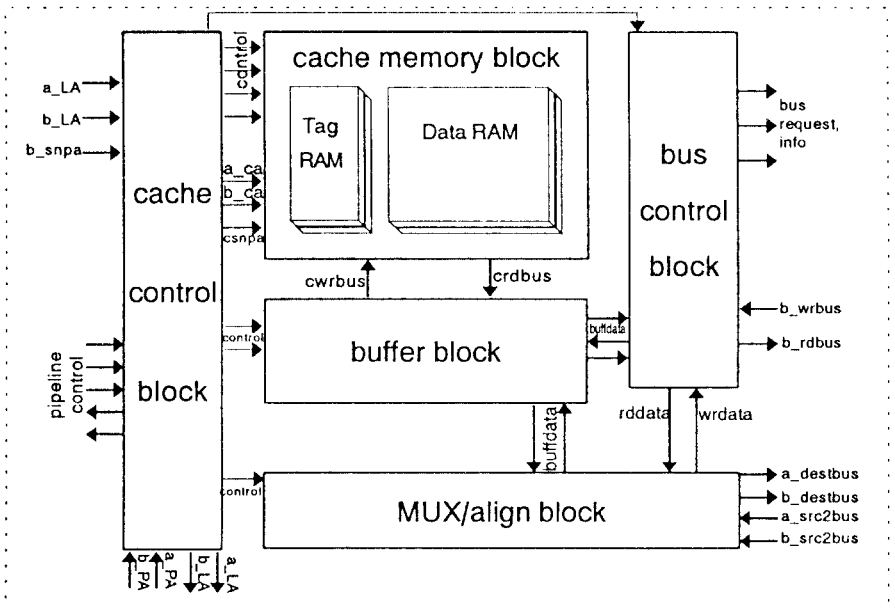

그림 2. 데이터 캐쉬 제어기의 구조
Fig. 2. Structure of the data cache controller

빈도로 사용된 라인을 교체한다. 교체할 라인이 M 상태인 경우는 그 라인을 교체 writeback 버퍼에 저장하고 writeback 을 시켜서 외부 메모리를 갱신 해줘야 한다. A-파이프 읽기/쓰기는 제어 유닛으로 부터 바이트 크기 정보(1,2,4 또는 8)를 받아서 캐쉬 엑세스를 한다. B-파이프 읽기/쓰기는 시퀘서로 부터 바이트 크기 정보(1,2 또는 4)를 받아서 캐쉬 엑세스를 한다. 이러한 사이클은 A-파이프 읽기/쓰기와 B-파이프 읽기/쓰기는 뱅크 충돌 및 캐쉬 미스가 일어나지 않으면 한 사이클에 동시에 처리가 가능하나, 플러시, 무효화, 라인 교체 사이클은 동시에 처리가 불가능하여, 여러 개의 사이클 요청이 동시에 들어오면, 우선권이 있는 사이클을 먼저 수행하게 된다.

마이크로프로세서가 캐쉬를 사용할 때, 메모리 데이터에 대하여 캐쉬와 캐쉬간, 캐쉬와 메모리 사이에 데이터 불일치 현상이 발생한다.[5] 본 논문에서는 현재 모토롤라 MC-88110, 인텔 i860 XP 등에 널리 사용되고 있는 MESI 프로토콜을 사용하였다.[6] MESI 프로토콜은 주로 멀티 프로세서 시스템 용도로 사용되고 있는나 분리된 캐쉬 구조를 갖는 단일 프로세서와 2차 캐쉬 사이에도 적용이 된다. MESI 프로토콜에서 정의된 4가지 상태는 다음의 표 1. 과 같다.

캐쉬에 저장된 정보의 상태를 나타내기 위해 각 캐쉬 라인은 MESI 상태중 한 상태를 갖는다. 캐쉬 라인 상태는 프로세서 읽기, 쓰기, 또는 기타 버스 마스터의 스누프 동작과 외부 핀 활동에 의해 천이가 된다. MESI 상태 천이에 대한 표는 표 2., 상태도는 그림 3. 과 같다.

2. 캐쉬 메모리 블럭

캐쉬 메모리 블럭은 8K 바이트 크기의 데이터 램(RAM)과 태그 램으로 구성이 된다. 캐쉬 메모리 블럭과 캐쉬 엑세스 주소에 대한 그림이 그림 4.에 나와있다. 캐쉬 라인의 크기는 32 바이트(256 비트)이며, 128 set이고, 2-way set associative 방식이다. 데이터 램은 8 개의 독립된 뱅크(bank)로 구성된 단일 포트 구조로 뱅크 충돌이 없으면 동시에 두 파이프에 데이터를 제공한다.[7]

수퍼스칼라 마이크로프로세서 캐쉬 메모리 구조로는 이중 포트(dual port) 구조와 단일 포트 구조가 있다. 이중 포트 구조는 Cyrix의 M1 프로세서에서 사용되는 방식이다. 두 구조의 비교는 표 3. 와 같다.

단일 포트 구조의 캐쉬 메모리는 6T(transistor) SRAM 셀을 사용한다. 이중 포트 구조의 SRAM 셀은 bit line 과 word line이 두 개씩이므로, 2 개의 트랜지스터가 추가로 필요하게 된다. 전체적인 면적으로 보면 단일 포트 구조가 이중 포트 구조보다 42% 적은 면적을 사용을 한다. 단일 포트 구조는 면적면에서는 유리하나, 주소를 선택하는 주소 multiplexer와 뱅크를 선택하는 데이터 multiplexer, 그리고 뱅크 충돌시 사이클을 serialize시키는 로직이 추가로 필요하게 된다.

뱅크 충돌은 A-파이프, B-파이프의 두 선형주소의

표 1. MESI 프로토콜의 4가지 상태
Table 1. Four states of the MESI protocol

M(Modified)	쓰기 히트에 의해 캐쉬 라인이 갱신되었음을 나타낸다. 갱신시 캐쉬는 시스템 버스를 스누프 하고, 스누프 라인에 히트가 발견되면 M 라인을 메인 메모리로 writeback시킨다.
E(Exclusive)	캐쉬 라인이 오직 한 캐쉬에만 존재하고, 내용은 메인 메모리와 같다.
S(Shared)	캐쉬 라인이 여러 캐쉬에 존재함을 나타낸다. 이 라인으로 쓰기 작업이 일어나면 write-through 사이클이 일어난다.
I(Invaid)	Reset후의 초기 상태로 캐쉬 라인이 무효하다는 것을 나타낸다.

표 2. 읽기, 쓰기, 스누프 사이클에서의 MESI 상태천이
Table 2. MESI state trasitions in read, write, snoop cycle
(a) 읽기 사이클에서의 MESI 상태천이

현재상태	pin cativity	다음상태	설명
M	관계 없음	M	읽기 히트, 버스 사이클 없음
E	관계 없음	E	읽기 히트, 버스 사이클 없음
S	관계 없음	S	읽기 히트, 버스 사이클 없음
I	CACHE#=0 & KEN#=0 & WB/WT#= & PWT=0	E	읽기 미스, 버스 읽기 사이클, 첫 BRDY# 또는 NA#에서 WB/WT# 가 1일때
I	CACHE#=0 & KEN#=0 & (WB/WT#= & or PWT=0)	S	읽기 미스, 버스 읽기 사이클, 첫 BRDY# 또는 NA#에서 WB/WT# 가 0일대
I	CACHE#=1 또는 KEN# = 1		KEN# = 1, 라인은 캐쉬에 저장되지 않음.

(b) 쓰기 사이클에서의 MESI 상태천이

현재상태	pin activity	다음 상태	설명
M	관계 없음	M	쓰기 히트, 캐쉬 내용 갱신, 버스 사이클 없음.
E	관계 없음	M	쓰기 히트, 캐쉬 내용 갱신, 버스 사이클 없음.
S	PWT=0 & WB/WT#=0	E	쓰기 히트, 캐쉬 내용 갱신, write-through 버스 사이클,
S	PWT=0 & WB/WT#=0	S	쓰기 히트, 캐쉬 내용 갱신, write-through 버스 사이클.
S	PWT=1	S	Write-through 페이지의 라인으로 쓰기 히트, write-through 버스 사이클
I	관계 없음	I	쓰기 미스, write-through 버스 사이클, Allocation 없음.

(c) 스누프 사이클에서의 MESI 상태천이

현재상태	다음상태 (INV=1)	다음상태 (INV=0)	설 명
M	I	S	M 라인으로 스누프 히트, writeback 버스 사이클, HIT#=0, HITM#=0
E	I	S	스누프 히트, 버스 사이클 없음, HIT#=0
S	I	S	스누프 히트, 버스 사이클 없음, HIT#=0
I	I	I	주소가 캐쉬에 없음, HIT#=0

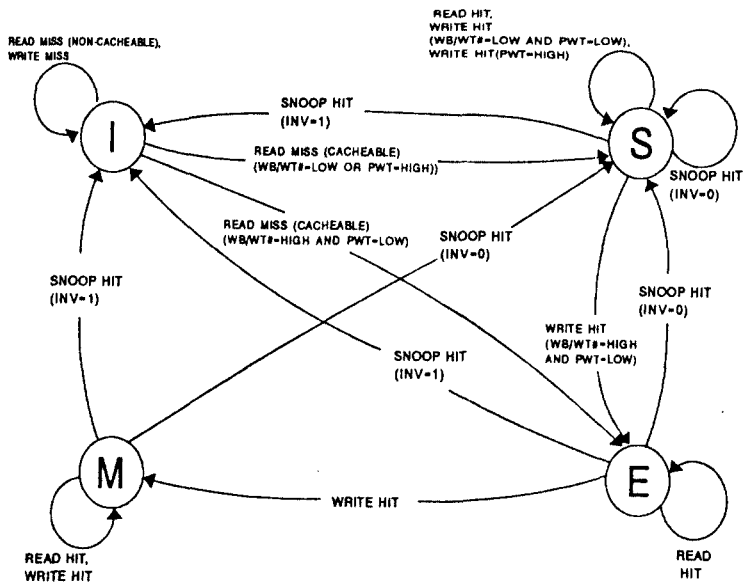

그림 3. MESI 상태 천이도
Fig. 3. State transition diagram of MESI state

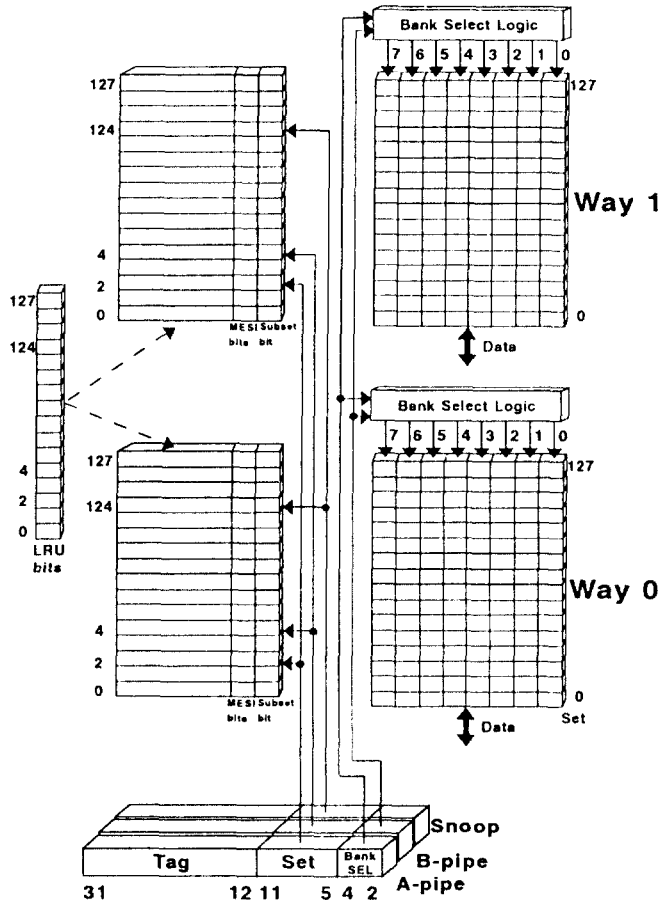

그림 4. 캐쉬 메모리 블럭의 구조
Fig. 4. Structure of the cache memory block

표 3. 이중 포트 구조와 단일 포트 구조의 비교
Table 3. Comparison of dual port structure and single port structure

기 능	이중 포트	단일 포트, interleabed 8뱅크 구조
면 적	2.0	1.0
전 력	2.0	1.0
set decoder 수	2	2
주소 mux 수	없음	8
데이터 mux 수	없음	1
뱅크 충돌 로직	없음	있음
제어의 복잡성	간단	복잡

address[4:2]가 같을때 발생한다. 뱅크 충돌 발생시, 캐쉬 제어기는 B-파이프의 사이클 정보를 latch시키고, A-파이프의 사이클을 먼저 수행한다. B-파이프 사이클은 A-파이프 사이클 종료후에 수행이 된다.

태그 램은 3-포트 구조로 A-파이프, B-파이프, 스누프의 3개의 태그 주소를 엑세스한다. 각 라인에는 캐쉬 데이터의 상태를 나타내는 LRU, MESI, subset, WP(write protect), 패리티(parity) 비트들을 두고 있다.[8] LRU 비트는 라인 교체 시에 어느쪽 way를 교체해야 하는지를 나타낸다. MESI 비트는 MESI 프로토콜에 의한 상태 비트를 저장하여 라인과 외부 메모리와의 관계를 표시한다. Subset 비트는 세그멘트 디스크립터 캐쉬와의 관계를 나타낸다. 디스크립터 내용을 읽는 경우 캐쉬 제어기는 이 비트를 set 시킨다. 캐쉬 라인 교체시 LRU 비트, MESI 비트, Subset 비트에 따라서 교체할 라인이 결정된다.

3. 캐쉬 버퍼 블럭

캐쉬 데이터 엑세스를 좀 더 효율적으로 하기 위한 부분으로, line fill 버퍼와 2개의 읽기 버퍼, 쓰기 버퍼와 3개의 WB(writeback) 버퍼로 구성이 된다. Line fill 버퍼는 캐쉬 읽기 미스가 일어나 line fill 동작이 일어날 때, 외부로부터 가져온 메모리 데이터를 이 버퍼에 써 준다. Line fill 동작이 끝나면 완전한 한 라인이 이 버퍼에 저장이 되고 캐쉬 메모리에 써 준다. 한편, line fill 작업 시에 line fill 버퍼에 있는 내용이 읽기 요청이 되면, 직접 버퍼에서 데이터를 내보낼 수 있다. 읽기 버퍼는 메모리 엑세스 주소가 정렬이 되지 않아 split 사이클이 발생될 때 사용이 된다. 쓰기 버퍼는 쓰기 미스가 일어나거나, S상태 라인으로 데이터를 써야 하는 경우, 즉 외부 메모리로 데이터를 내보내야 하는 경우 직접 버스 유닛으로 보내지 않고, 이 버퍼에 쓴다. 쓰기 버퍼링을 해주면, 실제 외부 메모리 쓰기 사이클이 종료되지 전에 캐쉬 제어기는 다른 사이클을 수행할 수 있다.[9] 쓰기 버퍼는 A-파이프용, B-파이프용의 두 버퍼가 존재하여 한 사이클에 두 버퍼에 써 주는 것이 가능하다. 쓰기 버퍼는 내용이 로딩이 되면, 버스 사용 요청을 하고, 승인(acknowledge)이 되면 버스 사이클을 실행하여 외부로 데이터를 내보낸다. 버스의 승인은 버스가 busy가 아닌 상태에서만 가능하므로 A-파이프, B-파이프 쓰기 버퍼가 동시에 버스 사이클을 수행할 수는 없고, 두 버퍼가 동시에 버스 요청을 하면 A-파이프 쓰기 버퍼가 우선권을 가져서 버스 사이클을 수행하고, 그 후에 B-파이프 쓰기 버퍼가 버스 사이클을 수행한다. WB 버퍼는 M상태 라인을 외부 메모리로 write-back시킬 때, 그 라인을 저장해 두는 버퍼로 무효화(invalidation) WB 버퍼, 내부 스누프 WB 버퍼, 교체(replacement) WB 버퍼를 설계하였다.

4. MUX/정렬 블럭

읽기 사이클일 때, 요청된 바이트 길이의 데이터를 캐쉬 라인으로부터 선택을 하고 정렬을 해준다. 읽기 동작일때의 데이터의 선택과정이 그림 5. 에 나와있다. 데이터를 읽을 때 8 바이트인 경우 b_destbus과 a_destbus, 4바이트인 경우에는 a_destbus로 데이터가 나가게 된다. B-파이프 읽기/쓰기는 1,2 또는 4 바이트의 데이터를 엑세스한다. 이 블럭에서는 a_destbus, b_destbus을 구동하기 전에 여러 작업을 해준다. 요청된 사이클이 4 바이트 이하인 경우에는 상위 2 바이트를 00 h 로 만들어 준다. 요청된 사이클이 1 바이트인 경우에는 a_destbus,b_destbus의 최하위 바이트(바이트 0) 에 데이터가 제공이 되고, 또한 바이트 1에도 동일한 바이트가 복제(duplicate)되어 실행 유닛의 32 비트 레지스터 AH,BH,CH,DH에 데이터를 제공한다. 쓰기 동작일때의 데이터의 흐름이 그림 6. 에 나타나있다. 데이터 쓰기 작업 시에는 8 바이트인 경우 b_src2bus ,a_src2bus, 4바이트 이하인 경우에는 a_src2bus이 사용된다.

5. 버스 제어 블럭

버스 제어 블럭은 캐쉬 미스가 일어나거나, 캐쉬 사이클 수행 중에 외부 메모리로 데이터를 내보낼 때에 캐쉬 제어기와 버스 유닛과 인터페이스를 한다. 이 블럭에서 요청하는 버스 사이클은 다음과 같다.

.외부 메모리 읽기 사이클
.외부 메모시 쓰기 사이클
.I/O 읽기 사이클
.I/O 쓰기 사이클

버스 제어 블럭은 버스 사이클이 필요하면 버스 유닛으로 요청을 한다. 버스 유닛은 버스 사이클 정보를 구

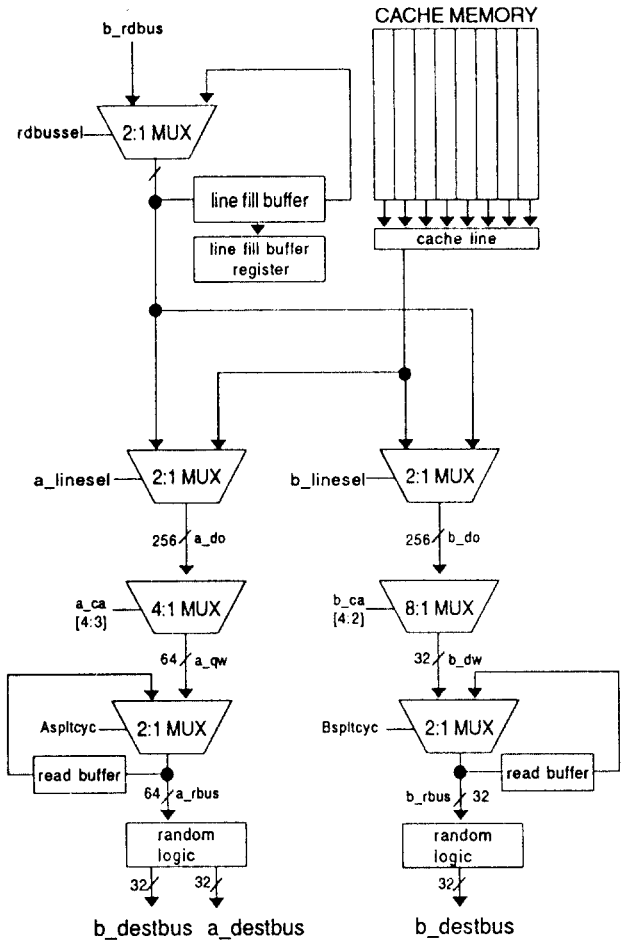

그림 5. 데이터 읽기 선택과정
Fig 5. Multiplexing of read data

동하라고 신호를 보내고, 버스가 busy가 아니면 승인을 보낸다. 이 순간부터 버스 사이클이 개시가 되어 외부 메모리의 데이터를 읽거나 데이터를 외부 메모리로 쓴다.

Ⅳ. 시뮬레이션 및 결과

본 연구에서는 펜티엄 마이크로프로세서 데어터 캐쉬 제어기를 Verilog-HDL를 사용하여 모델링을 한 다음, 테스트 벡터를 입력으로 가하여 동작 시뮬레이션을 행하였다. 데이터 캐쉬 제어기의 각 블럭을 모듈로 작성을 한 다음, 각 모듈을 합하여 하나의 모듈로 만들고, 테스트 벡터 모듈로 동작의 시뮬레이션을 하였다.

1. 읽기 동작

읽기 동작에 대한 데이터 캐쉬 제어기의 시뮬레이션은 그림 7. 과 같다.테스트 벡터로 A-파이프, B-파이프 각각 3 개씩의 선형주소를 입력으로 하였다. 파형을 보면 두 파이프 모두 읽기 미스, A-파이프 읽기 히트 - B-파이프 읽기 미스, 두 파이프 모두 읽기 히트에 대한

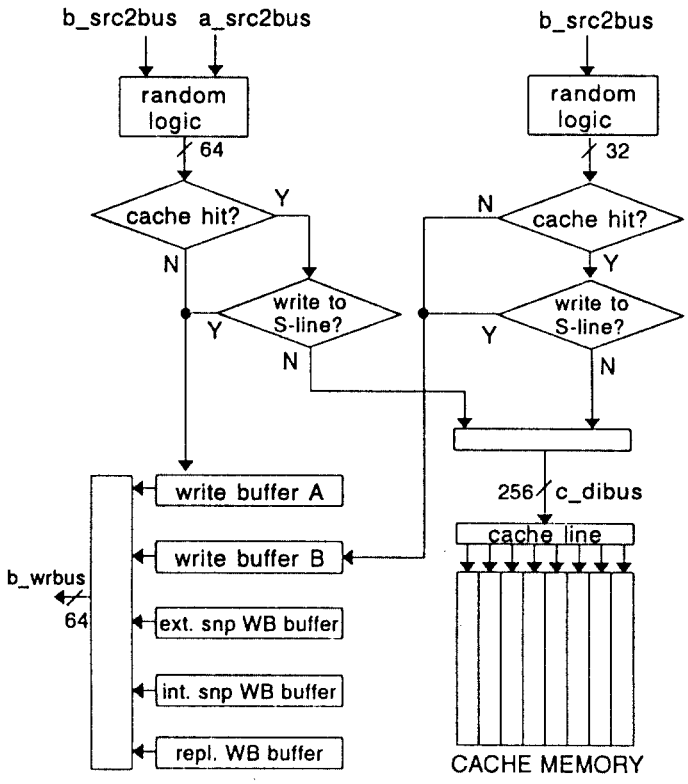

그림 6. 데이터 쓰기 흐름
Fig 6. Flow of write data

과정이 나타나 있다. 읽기 미스가 일어났을 때, 캐쉬 제어기는 버스 유닛에 버스 사이클 요청을 하여, 외부 메모리로 부터 데이터를 가져오는 line fill 동작을 해준다. 버스 유닛이 busy가 아니고, 외부 메모리 엑세스시 대기 상태가 없다고 보면 버스 요청후 3 사이클 후에 data request first 방식으로, 라인 버퍼로 외부 메모리 데이터가 들어온다. 캐쉬 제어기는 라인 버퍼의 내용에서 요청된 데이터를 선택하여 a_destbus, b_destbus에 데이터를 제공한다.

2. 쓰기 동작

쓰기 동작에 대한 시뮬레이션이 그림 8. 에 나와있다. 테스트 벡터로는 A-파이프, B-파이프 각각 3개씩의 선형주소를 입력으로 하였다. 두 파이프 모두 쓰기 미스, A-파이프 쓰기 히트-B-파이프 쓰기 미스, 두 파이프 모두 쓰기 히트에 대한 파형이 그림에 나타나있다. 쓰기 미스 발생시, 데이터는 직접 버스 유닛으로 나가지 않고, 쓰기 버퍼로 가게 된다. 쓰기 버퍼는 background 작업으로, 버스 유닛에 사이클을 요청을 한다. 쓰기 히트 발생시, 캐쉬 제어기는 그 라인의 MESI 상태 비트에 따라 M 상태 또는 E 상태인 경우, MESI 비트를 M 으로 변경하여 사이클을 끝내고, S 상태 라인인 경우에는 그 라인을 write through시킴을 알 수 있다.

3. Writeback 동작

캐쉬 읽기 미스가 일어나면 캐쉬는 line fill 동작을 하여, 한 라인을 읽어온다. 캐쉬는 그 라인을 캐쉬 메모

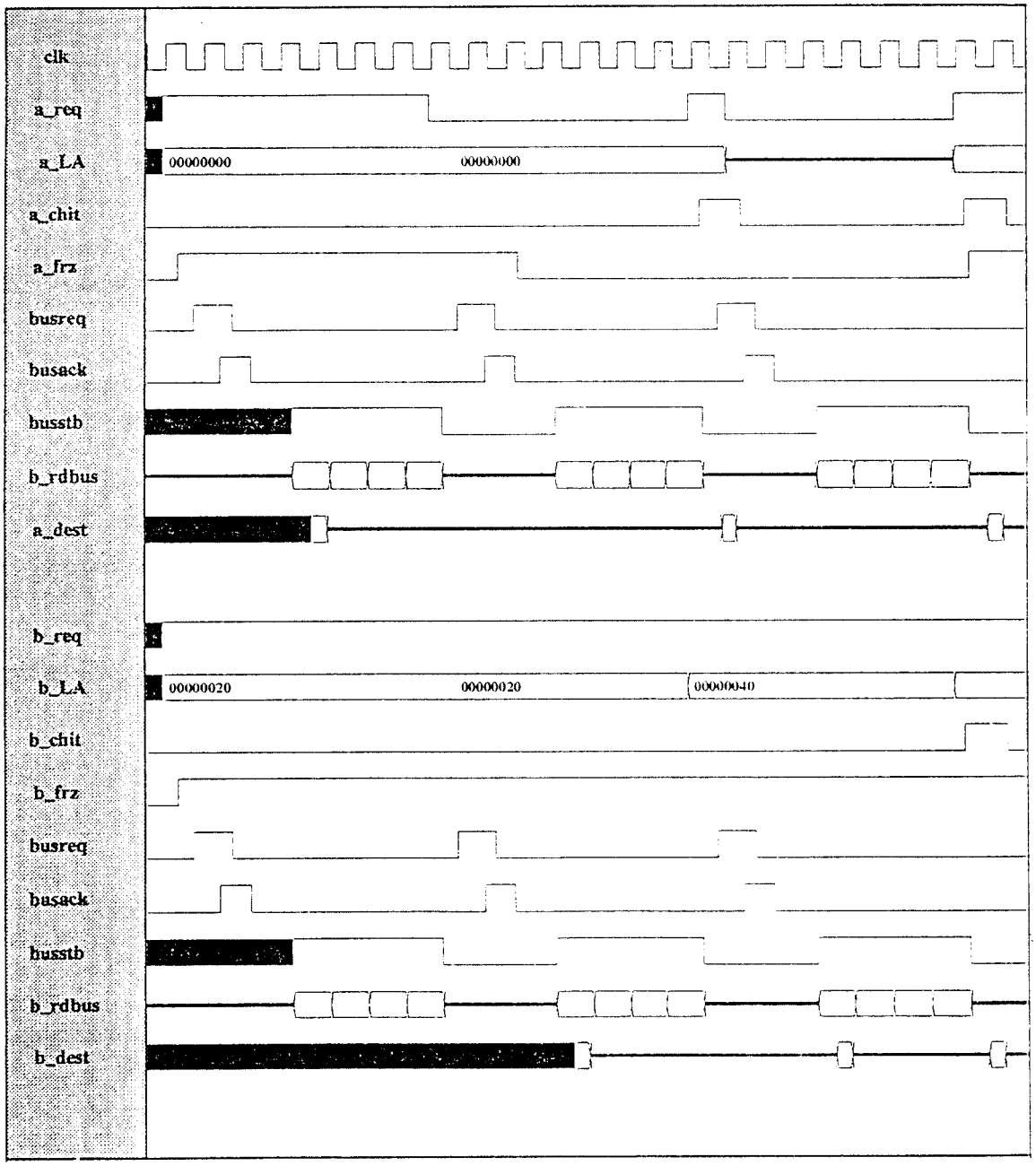

그림 7. 캐쉬 읽기 동작에 대한 시뮬레이션
Fig 7. Simulation of the cache read cycle

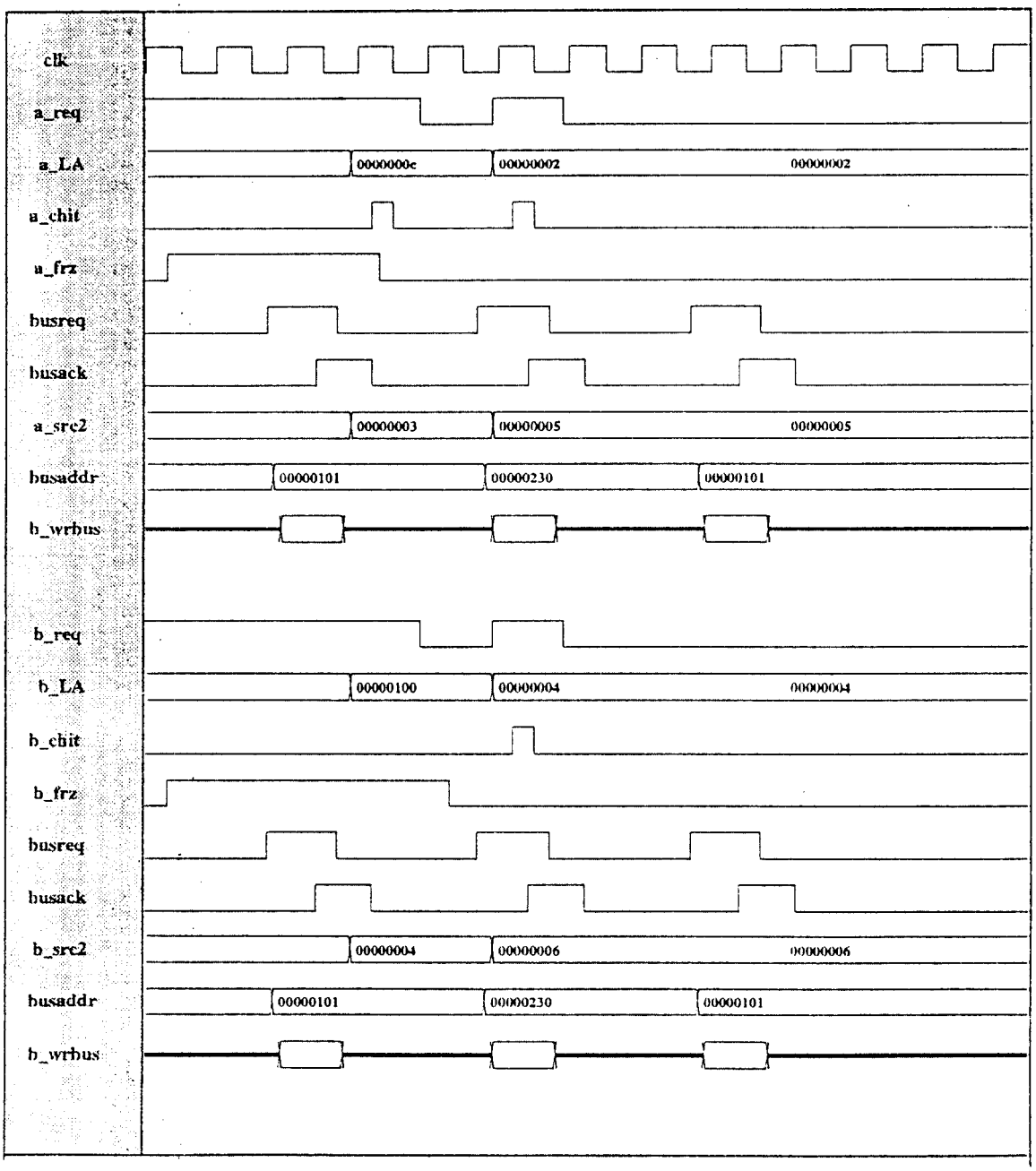

그림 8. 캐쉬 쓰기 동작에 대한 시뮬레이션
Fig 8. Simulation of the cache write cycle

리에 써줘야 하는데, 캐쉬 메모리에 있는 라인의 상태가 M인 경우 그 라인은 유효한 데이터이므로 외부 메모리로 writeback을 시켜줘야만한다. 그 과정에 대한 시뮬레이션은 그림 9.과 같다..

4. 데이터 캐쉬 제어기 성능 측정

본 연구에서 제안하고 HDL로 모델링한 수퍼스칼라 데이터 캐쉬 제어기의 성능을 측정하기 위하여 응용 프로그램을 사용하여 캐쉬 적중률(hit ratio)을 측정하였다. 성능 측정을 위하여 응용 프로그램을 데이터 캐쉬 제어기의 시뮬레이션 입력 테스트 벡터로 하는 어드레스 트레이스 드리븐(address trace-driven) 시뮬레이션을 행하였다. 응용 프로그램으로부터 입력 테스트 벡터는 그림 10.과 같은 순서로 작성하였다.

성능 측정 결과는 다음 표 4. 와 같다.

5. 성능 향상도

표 5.는 기존의 writeback 캐쉬 제어기 i486 내부 캐쉬 제어기와 본 연구에서 제안한 수퍼스칼라 캐쉬 제어기의 성능을 비교한 것이다. 기존의 단일 파이프라인 구조에서 캐쉬 제어기는 1 클럭 사이클에 하나의 데이터를 엑세스하면 충분하였으나, 독립된 두 파이프라인을 갖는 수퍼스칼라 마이크로프로세서에서는 1 클럭 사이클에 두개의 데이터 엑세스가 요구된다. 제안된 펜티엄 마이크로프로세서 캐쉬 제어기는 캐쉬 히트시 데이터 엑세스 측면에서기존의 writeback 캐쉬 제어기보다 2배의

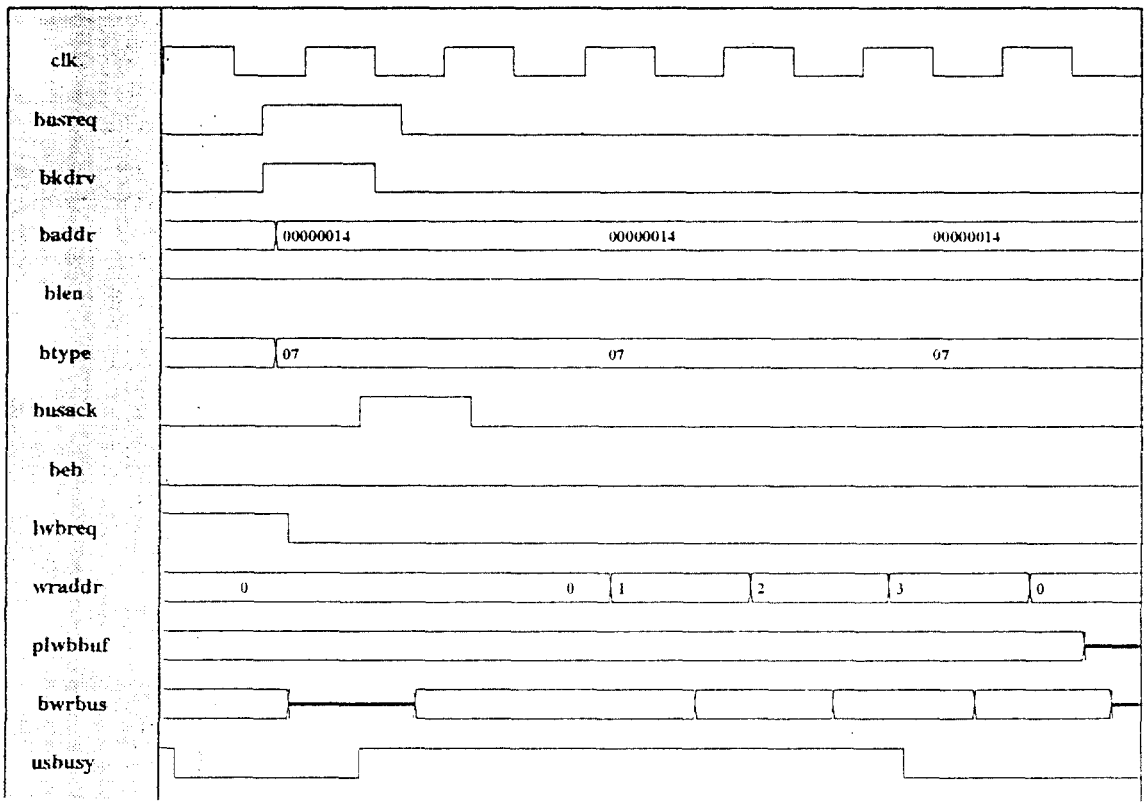

그림 9. Writeback 동작에 대한 시뮬레이션
Fig 9. Simulation of the writeback cycle

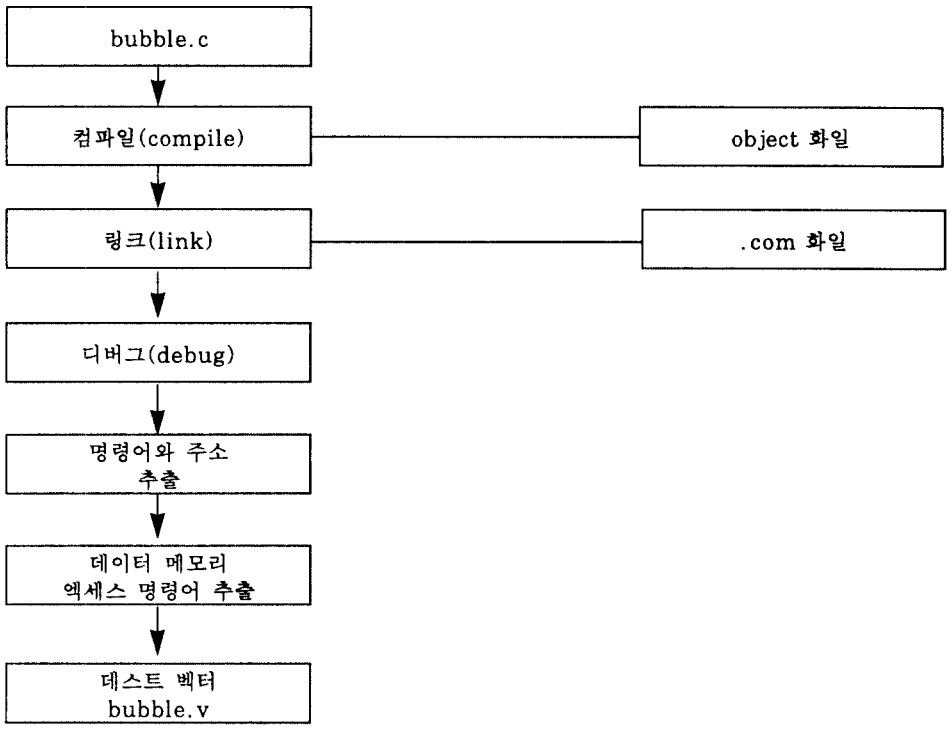

그림 10. 테스트 벡터 작성 순서도
Fig. 10. Flow-chart of generationg test vector

표 4. 성능 측정 결과
Table 4. Result of the performance evaluation

C 언어 소스 프로그래명	bubbed.c
소스 프로그램 크기	17 라인
A-파이프 메모리 읽기 요청 수	231
A-파이프 메모리 쓰기 요청 수	106
B-파이프 메모리 읽기 요청 수	168
B-파이프 메모리 쓰기 요청 수	36
총 메모리 요청 수	539
A-파이프 캐쉬 히트 횟수	336
B-파이프 캐쉬 히트 횟수	201
총 캐쉬 히트 횟수	537
캐쉬 적중률	99%

표 5. 성능 향상도
Table 5. The measure of performance improvement

캐쉬 제어기 / 동작	i486용 내부 캐쉬 제어기	제안된 펜티엄 내부 캐쉬 제어기
2개의 읽기 동작 (읽기 히트)	2	1
2개의 읽기 동작 (읽기 미스)	12 이상	11 이상
2개의 쓰기 동작 (쓰기 히트)	2	1
2개의 쓰기 동작 (쓰기 미스)	2*	1*

성능 향상을 가져왔다.

Ⅴ. 결론

본 연구에서는 한 사이클에 두개의 데이터를 엑세스하여 펜티엄 마이크로프로세서에 적합한 데이터 캐쉬 제어기의 구조를 제안하고 이를 Verilog-HDL를 사용하여 기능수준으로 기술을 하여, 제어기의 동작을 검증하였다. 수퍼스칼라 마이크로프로세서에서는 디코더, 선형 주소 발생기, ALU 등이 파이프라인의 갯수만큼 존재하여 각각 독립적으로 동작한다. 각 파이프라인이 동시에

캐쉬 메모리 요청을 하는 경우 캐쉬 제어기는 동시에 데이터를 엑세스 해야한다. 캐쉬 메모리를 단일 포트, 8개의 뱅크 구조로 설계하여 뱅크 충돌이 없는 경우에는 한 사이클에 두개의 데이터를 엑세스 가능하도록 하였다. 이러한 뱅크 구조의 캐쉬는 좀 더 확장을 시키면, 두 파이프 이상의 수퍼스칼라 마이크로프로세서에도 적용이 가능하다. 버스의 이용횟수를 최소화 시키기 위해

writeback 기법을 사용하였고, 또한 외부 메모리 쓰기 사이클을 줄이기 위해 쓰기 버퍼를 사용하였다. 캐쉬의 일관성 유지를 위해서는 스누핑과 MESI 프로토콜을 채택하였다. 앞으로 21세기에는 고집적 기술에 의해 1 Mbyte급의 데이터 캐쉬 메모리가 내장될 예정이다. 이러한 대용량의 캐쉬에서는 면적면에서 유리하고, 다수의 파이프라인에 동시에 데이터를 제공하는 interleaved 단일 포트 구조의 캐쉬가 적합하므로, 이에 대한 체계적인 연구가 필요하다.

參 考 文 獻

1. Yong S. Lee, "A Secondary Cache Controller Design for a High-End Microprocessor," IEEE Journal of Solid-State Circuits, Vol. 27, No. 8, pp.1141-1146, August. 1992.
2. Jim Handy, *The Cache Memory Book*, Academic Press, pp.60-62, 1993.
3. Donald Alpert, "Architecture of the Pentium Microprocessor", IEEE Micro, pp.14-15, June 1993.
4. Pentium Processor User's Manual, Vol. 1, Intel, pp.3-1,3-3, 1993.
5. James Archibald, Jean-Loup Baer, "Cache Coherence Protocols: Evaluation Using a Multiprocessor Simulation Model, ACM Transactions on Computer Systems", Vol. 4, No. 4, pp.339-341, Nov. 1986.
6. Keith Diefendoriff, Michael Allen, "Organization of the Motorola 88110 Superscalar RISC Microprocessor", IEEE Micro, pp.56-59, Apr. 1992.
7. Andrew Wolfe, Rodney Boleyn, "Two-ported Cache Alternatives for Superscalar Processors", Proceedings of the 26th Annual International Symposium on Microarchitecture, pp.43-44, Dec. 1993.
8. Don Anderson, Tom Shanley, *Pentium Processor System Architecture*, MindShare Press, pp.167-171, 1993.
9. Nikitas Alexandridis, *Design of Microprocessor-Based Systems*, Prentice Hall, p.257, 1993.

李 鎔 錫(Yong Surk Lee) 정회원
1950년 10월 23일생
1969년 3월~1973년 2월 : 연세대학교 전기공학과(공학사)
1975년 3월~1977년 2월 : Michigan 반도체설계 공학석사
1977년 3월~1981년 12월 : Michigan 반도체설계 Ph.D
1982년 3월~1983년 12월 : Sperry-Univac Computer 설계 엔지니어
1984년 1월~1986년 9월 : Hyundai Electronics, USA 설계 엔지니어
1986년 10월~1988년 8월 : National Semiconductor 설계 엔지니어
1988년 9월~1989년 8월 : Performance Semiconductor 설계 엔지니어
1989년 9월~1992년 12월 : Intel Corporation 설계 엔지니어
1993년 3월~현재 : 연세대학교 전자공학과 부교수