혼합 지연 모델에 기반한 비동기 명령어 캐시 설계

Design of an Asynchronous Instruction Cache based on a Mixed Delay Model

전광배*, 김석만*, 이제훈**, 오명훈***, 조경록*
충북대학교 정보통신 공학과*, 강원대학교 전자공학과**, 한국전자통신연구원***

Kwang-Bae Jeon(kbjeon@hbt.cbnu.ac.kr)*, Seok-Man Kim(smkim@hbt.cbnu.ac.kr)*,
Je-Hoon Lee(Jehoon.lee@kwangwon.ac.kr)**, Myeong-Hoon Oh(mhoonoh@etri.re.kr)***,
Kyoung-Rok Cho(krcho@cbnu.ac.kr)*

요약

최근에는 프로세서의 고성능화에 따라 명령어 캐시와 데이타 캐시를 분리하는 구조의 설계가 일반적이다. 본 논문에서는 혼합 지연모델을 갖는 비동기식 명령어 캐쉬구조를 제안하며, 데이타 패스에는 지연무관인 회로모델을 적용하고 메모리 에는 번들지연모델을 도입하였다. 요소기술로는 명령어 캐시는 CPU, 프로그램 메모리와 4-상 핸드쉐이크(hand-shake) 프로토콜로 데이터를 전달하고, 8-K바이트, 4상 연관의 맵핑 구조를 가지며 Pseudo-LRU 엔트리 교체알고리즘을 채택하였다. 성능분석을 위하여 제안된 명령어 캐시를 게이트레벨로 합성하고 32비트 임베디드 프로세서와 연동하는 플랫폼을 구축하였다. 구축한 플랫폼에서 MI벤치마크 프로그램을 테스트하여 99%의 캐시히트율과 레이턴시가 68% 감소하는 결과를 얻었다.

■ 중심어 : | 비동기 | 캐시메모리 | 비동기 회로 | 캠 |

Abstract

Recently, to achieve high performance of the processor, the cache is splits physically into two parts, one for instruction and one for data. This paper proposes an architecture of asynchronous instruction cache based on mixed-delay model that are DI(delay-insensitive) model for cache hit and Bundled delay model for cache miss. We synthesized the instruction cache at gate-level and constructed a test platform with 32-bit embedded processor EISC to evaluate performance. The cache communicates with the main memory and CPU using 4-phase hand-shake protocol. It has a 8-KB, 4-way set associative memory that employs Pseudo-LRU replacement algorithm. As the results, the designed cache shows 99% cache hit ratio and reduced latency to 68% tested on the platform with MI bench mark programs.

■ keyword : | Completion Signal | Cache Memory | Delay-Insensitive | CAM |

I. 서 론

소형 및 휴대용기기에 많이 사용되는 임베디드 프로세서는 소비전력을 줄이는 것이 중요한 이슈이며, 클럭을 사용하지 않는 비동기 프로세서 설계 기법은 소비전력을 줄이는 설계방법중의 하나로 볼 수 있다. 최근에

* 본 연구는 교육과학기술부와 한국산업기술재단의 지역혁신인력양성사업으로 수행된 연구결과임
접수번호 : #100126-003
접수일자 : 2010년 01월 26일
심사완료일 : 2010년 03월 22일
교신저자 : 조경록, e-mail : krcho@chungbuk.ac.kr

임베디드 시스템의 고성능화로 프로세서의 성능향상을 위해 캐시의 사이즈를 증가 시키는 것이 보편화되고 있다. 그러나 캐시는 액세스 빈도가 높고 SRAM 으로 설계되어 사이즈가 커질수록 더욱 많은 에너지 소비가 발생하게 되며, 프로세서 전체에서 소비하는 에너지 중에서 캐시에서 소비되는 에너지의 비중이 점점 더 증가된다. StrongARM의 경우 총 에너지 소모율 중 캐시 참조로 인한 에너지 소모율이 40%를 차지한다[1]. 본 논문에서는 캐시의 저전력화 구현을 위해 전역 클럭 신호를 이용하지 않는 비동기식 명령어 캐시의 구조를 제안한다. 제안된 캐시는 주소 태그(Tag) 저장을 위하여 램(RAM)이 아닌 내용 연관기억 장치인 캠(Contents Addressable Memory)으로 구현 하였다

본 논문에서 제안한 비동기식 명령어 캐시는 지연을 인식하지 않는 지연무관(Delay-Insensitive) 모델과 번들(Bundled) 지연방식 모델을 동시에 적용한 혼합 지연 모델로 설계 하였다. 설계된 캐시의 동작은 다음과 같다. 캐시가 CPU로부터 요청된 주소를 가지고 있을 경우 match 신호가 high가 된다. high로 천이된 match 신호는 캠의 동작 완료 시간을 나타내므로 지연무관 모델로 동작 할 수 있게 되고, high로 천이된 match 신호는 hit 신호를 발생시켜 캐시 히트임을 알려준다. 그러나 CPU로부터 요청된 주소가 캠에 없을 경우 완료 신호를 생성할 수 없게 된다. 캠의 match 신호는 초기에 low인 상태에서 입력되는 태그와 저장된 태그를 비교하여 같으면 match 신호가 high로 되지만, 캠의 내부에 입력되는 태그와 일치하는 태그가 없을 경우를 match 신호를 high로 천이 시키지 못한다. 이를 미스히트라 하고, 이때 match 신호는 계속 low인 상태를 유지하게 된다. 즉, 캐시는 동작 완료 시간을 생성하지 못하므로 미스히트 처리는 고정 지연을 통하여 요청신호를 출력으로 보내는 번들지연방식을 사용한다. 번들지연은 캐시 고유의 동작지연보다 큰 지연소자를 사용하고 프로그램 메모리로부터 새로운 명령어를 요청하여 캐시에 업 로드하는 동작을 수행 하게 한다. 제안된 명령어 캐시는 UMC 0.13um 라이브러리를 사용하여 합성한 후 비동기식 32비트 EISC 프로세서[2]와 연동하여 MI 벤치마크[3] 프로그램을 실행시켜 성능을 평가 하였다.

본 논문의 구성은 Ⅱ장에서 제안된 비동기식 명령어 캐시의 구조와 캐시 히트와 캐시 미스히트에 따른 동작과 번들지연방식을 위한 고정 지연 소자의 구성에 대하여 기술하고 Ⅲ장에서는 캠 셀의 구조와 match 신호의 발생 대하여 기술하였다. Ⅳ장에서는 비동기식 32비트 EISC 프로세서와의 연동 테스트 플랫폼과 MI 벤치마크 프로그램을 이용하여 명령어 캐시의 성능을 평가 하였다.

Ⅱ. 비동기 명령어 캐시 구조

제안된 비동기식 명령어 캐시는 32비트 임베디드 프로세서용으로 설계되었으며 [그림 1]과 같이 태그를 저장하는 캠, 명령어를 저장하는 램, CPU와 프로그램 메모리와의 핸드 쉐이크 및 캐시 제어를 위한 캐시컨트롤러로 구성된다. 또한, 캐시컨트롤러에는 캠과 램의 번들 지연 동작을 위한 지연 회로 $\triangle t0$, $\triangle t1$, 가 포함되어 있다. 캠은 I_Req 신호가 high로 인에이블 되면 주소버스 데이터 중 태그와 캠에 저장된 태그를 비교하여 일치하면 hit 신호를 high로 인에이블 시켜 캠의 동작이 완료되었음을 캐시 컨트롤러로 알려 주고 동시에 주소의 인덱스를 램으로 보낸다. 램은 I_Req 신호가 high로 천이하고 캠으로부터 인덱스를 받으면 해당 인덱스에 매핑되어 있는 명령어를 출력한다. 캠에 일치하는 태그가 없으면 캠은 I_Req 신호가 high로 인에이블되고 $\triangle t0$ 후에도 hit 신호가 high로 인에블되지 않고 캐시는 미스히트라 판단하여 캐시의 miss 신호는 P_Req를 high로 인에이블시켜 프로그램 메모리에 새로운 명령어를 요청한다. 새로운 명령어를 요청한 후 프로그램 메모리의 완료 신호인 P_Ack 신호가 high로 천이하면 명령어 캐시는 주소 버스에서 태그는 캠에, 새로운 명령어는 램에 저장한다. 새롭게 저장된 태그는 히트가 되게 하고, 램 지연 시간 $\triangle t1$ 후에 I_Ack 신호를 인에이블 시켜 명령어 캐시의 동작 완료를 CPU로 알려준다. 설계된 명령어 캐시는 I_Req, I_Ack, P_Req, P_Ack를 사용하여 CPU 및 프로그램 메모리와 각각 핸드 쉐이크 프로토콜로 통신한다.

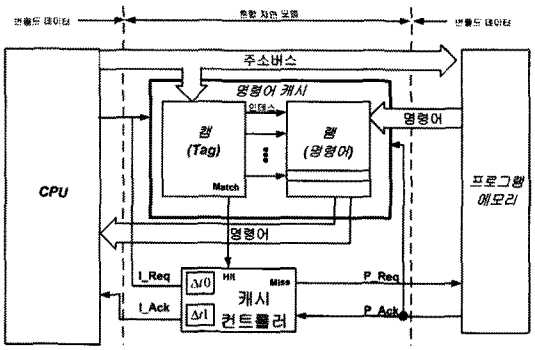


그림 1. 제안된 비동기식 명령어 캐시

1. 명령어 캐시 정책

제안된 명령어 캐시는 구조적으로 4상 연관 맵핑 방식을 가지고 8-K바이트의 크기로 설계 되었다. 효율적인 캐시 관리를 위해 엔트리 교체 알고리즘으로 Pesudo-LRU 방식을 사용하였다. 또한 캐시 초기화시에 캠의 모든 셀들이 high나 low 상태로 초기화 되면, 초기화된 상태와 요청된 주소가 일치 할 경우 정확한 동작을 할 수 없게 된다. 설계된 캐시는 초기화에 따른 에러를 피하기 위해 캠의 각 태그는 상태비트(V)를 가지도록 설계 하였다. [그림 2]는 명령어 캐시의 동작을 나타낸다. CPU로부터 전달된 32비트의 주소를 각각 태그, 인덱스, 오프셋으로 분리한다. 태그의 상태비트가 유효한 상태를 나타내지 않으면 캠은 hit 신호를 인에이블 시키지 않고 miss 신호를 인에이블 시키게 된다. 태그는 캐시에 CPU에서 요청한 명령어가 저장되어 있는가를 판단하고, 인덱스는 요청한 명령어가 포함된 블럭의 위치를 알려준다. 오프셋은 요청된 명령어가 포함된 블록 중 해당 워드를 선택하게 한다. 캐시에서 미스히트가 발생할 경우 miss 신호가 high로 인에이블 되며 프로그램 메모리로 새로운 명령어를 요청한다. 설계된 명령어 캐시는 효율적인 엔트리 교체를 위해 Pesudo-LRU 알고리즘을 사용하기 위하여 캐시를 4개의 영역으로 분리되어 관리한다. 캐시의 4개의 영역 중 어느 영역에서 히트가 나는지를 확인하여 히트가 가장 나중에 발생했던 영역에 새로운 명령어와 주소를 저장하도록 한다.

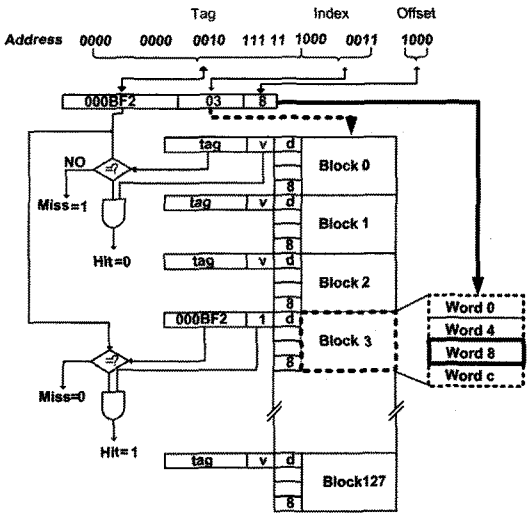


그림 2. 명령어 캐시의 동작

2. 번들지연모델 지연 회로

번들모델에 기반한 캐시 동작을 위하여 2개의 고정 지연 소자 $\Delta t0$, $\Delta t1$를 갖는다. $\Delta t0$은 캐시의 미스히트 판단을 위한 고정 지연 소자이고 $\Delta t1$ 는 램의 동작 완료를 위한 고정 지연 소자 이다. 고정 지연 회로의 구조는 [그림 3]과 같이 AND 게이트를 인버터로 사용하여 입력신호를 지연 시키는 체인 구조이다. 지연 체인의 게이트 수를 조정하여 필요한 지연을 얻게 된다. [표 1]과 [표 2]에는 캠과 램에 각각 필요한 AND 게이트의 수와 동작 완료 신호의 시간과 회로 설계에 적용된 시간의 비율을 프로세스 조건별로 합성하여 시뮬레이션 한 결과이다. 회로의 동작 완료 시간과 고정 지연의 시간의 차가 적을수록 회로의 동작은 빨라진다. 그러나 회로의 안정적인 동작을 위하여 일정 시간 이상 여유를 설정해야 하며, 본 논문에서는 번들지연은 회로의 동작 완료 시간의 10% 정도의 여유를 두고 설정 하였다.

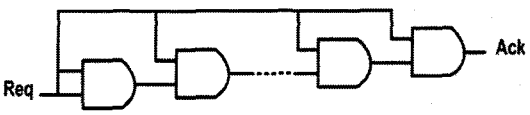


그림 3. 고정 지연을 위한 게이트 체인 회로

표 1. 캐시내 캠의 지연, $\triangle t0$.

CAM delay of each case(ns)		# of gate	Margin(%)	Designed delay(ns)
BC	0.604	22	10.75	0.669
TC	0.642	15	15.73	0.743
WC	1.014	17	16.57	1.182

표 2. 캐시내 메모리 램의 지연, $\triangle t1$.

RAM delay of each case(ns)		# of gate	Margin(%)	Designed delay(ns)
BC	0.848	15	14.01	1.103
TC	0.968	14	13.94	1.094
WC	1.824	17	12.61	2.054

3. 명령어 캐시 동작

3.1 명령어 캐시의 상태도

[그림 4]와 같이 캐시의 초기 상태는 Invalid한 상태로부터 시작한다. CPU로부터 동작요구 신호를 받고 캐시 미스히트 상황이 발생하여 프로그램 메모리로부터 명령어를 캐시에 저장하기 전에는 캠이 가지고 있는 데이터들은 태그가 아닌 Invalid 한 값이기 때문이다. Invalid상태에서는 CPU가 명령어를 요청하면 캐시 미스가 발생하고 프로그램 메모리로부터 유효한 명령어가 캐시에 저장될 때까지 Waiting 상태로 천이한다. 유효한 명령어가 프로그램 메모리로부터 캐시에 전달이 되면 Instr Valid 상태로 천이 한다. Instr Valid 상태에서는 CPU로부터 요청된 명령어가 캐시에 있으면 계속 Valid한 상태를 유지한다. 그러나 캐시에 없는 새로운 명령어를 요청하게 되면 유효한 명령어가 캐시로 전송될 때 까지 Waiting 상태에서 새로운 명령어를 기다리게 된다. 캐시가 동작을 시작하면 Invalid 상태로 초기화되기 전까지는 Instr Valid 상태를 유지한다.

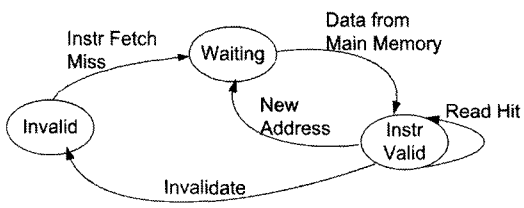


그림 4. 명령어 캐시의 상태도

3.2 명령어 캐시 히트 동작

제안된 명령어 캐시의 히트시 동작 타이밍은 [그림 5]와 같다. 먼저 CPU로부터 캐시에 명령어 요청은 주소 버스에 유효한 주소가 실린 후 I_Req 신호가 high로 천이되는 시점에서 인에이블 된다. 캠은 주소 버스의 태그와 캠에 저장된 태그를 비교한다. CPU에서 요청한 주소가 캠에 저장되어 있다면 캠의 hit 신호가 high로 천이되어 해당 명령어가 캐시에 저장되어 있음을 캐시 컨트롤러에게 알려준다. 이때 캐시 컨트롤러는 hit의 인에이블 시점을 캠의 동작 완료로 판단하고 캐시에서 직접 명령어가 출력 될 수 있도록 램을 인에이블 시킨다. 램에서 명령어가 출력될 때까지의 지연시간 $\triangle t1$ 후에 I_Ack 신호를 high로 인에이블 시켜 캐시의 동작 완료를 CPU로 알려 캐시와 CPU사이의 명령어 버스에 명령어가 유효함을 알려 주게 된다. I_Ack신호를 감지한 CPU는 명령어 버스의 명령어를 읽어가고 I_Req 신호를 low로 디스에이블 시키면 캐시는 모든 동작을 멈추고 다음의 I_Req 신호가 high로 인에블되기 전 까지 대기 상태를 유지하게 된다. 캐시 히트시에는 캠의 hit 신호를 이용하여캠의 동작 완료 시점을 알 수 있기 때문에 지연무관 모델에 기반한 비동기식 동작을 한다.

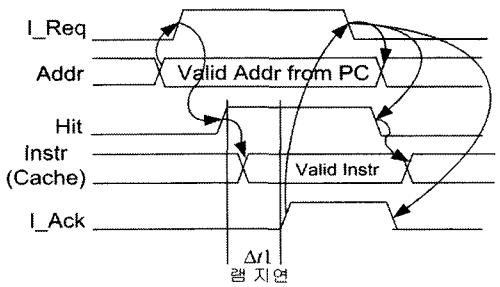


그림 5. 명령어 캐시 히트 타이밍 다이어그램

3.3 명령어 캐시 미스히트 동작

[그림 6]은 명령어 캐시 미스히트 동작 타이밍을 나타낸다. CPU에서 I_Req 신호를 high로 인에이블 시키면 캐시는 입력되는 주소의 태그 부분을 캠에 저장된 태그와 비교하게 된다. I_Req 신호가 인에이블 되고 나서 지연$\triangle t0$ 후에도 캠의 동작 완료 신호인 hit 신호가 low를 유지 하고 있으면 캐시 컨트롤러는 미스히트라

판단한다. 캐시 컨트롤러는 프로그램 메모리로 P_Req 신호를 인에이블 시켜 현재 주소 버스에 해당하는 명령어를 요청하게 된다. 프로그램 메모리에서 명령어가 출력되면 P_Ack를 high로 인에이블 시키게 된다. P_Ack 신호는 캐시의 캠과 램에 쓰기 신호를 인에이블 시켜 캠에는 주소버스 상의 태그가 에는 프로그램 메모리에서 출력되고 있는 명령어를 저장한다. 이 동작이 완료되면 캠에 현재 주소의 태그가 쓰여져 있으므로 hit 신호가 high로 인에이블 된다. 캐시가 히트상태가 되었으므로 램 지연시간 $\triangle t1$후에 I_Ack 신호를 high로 인에이블 시켜 캐시의 동작 완료를 CPU로 알려주게 된다. 즉 캐시 미스히트시에는 [그림 3]의 게이트 체인으로 $\triangle t0$과 $\triangle t1$ 을 도입하여 현재의 동작 완료를 판단하는 번들지연 방식의 모델을 이용 하였다[4][5].

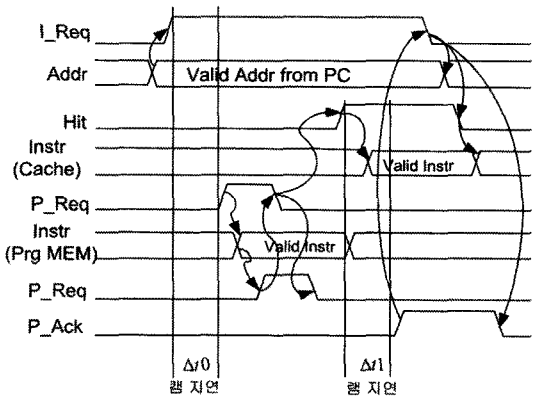


그림 6. 명령어 캐시 미스히트 타이밍 다이어그램

III. 명령어 캐시 설계

1. 캠의 구조

캐시 메모리는 다른 메모리에 비하여 고속 동작 이 요구 되므로 일반 적인 위치 참조 메모리보다 기억된 내용의 일부분만으로 내용을 유추 할 수 있는 연상 메모리인 캠이 사용된다. 캠의 모든 비트 셀은 비교기 로직을 가지고 있으며 태그 21비트를 비교 할 수 있도록 [그림 7]과 같이 어레이 구조로 되어 있다.

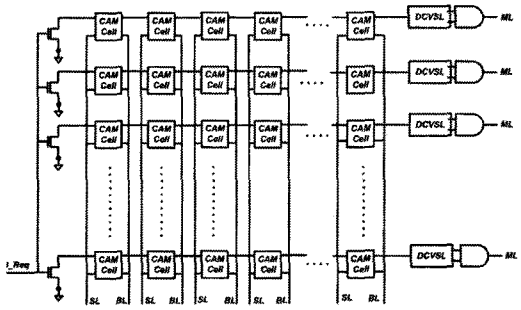


그림 7. 설계된 캠의 구조

2. DCVSL을 이용한 캠 연산완료 신호

설계된 비동기식 캠은 클럭을 사용하지 않으므로 각 회로블럭의 동작완료 신호를 발생시켜 회로가 다음 동작을 할 수 있는 상태로 천이토록 한다. 본 논문에서는 DCVSL 회로를 이용하여 동작완료 신호를 얻는다. [그림 8]은 NMOS 트리와 두 개의 차동 출력을 갖는 DCVSL의 구조로 캠셀의 비교연산 완료 신호를 생성하는 회로이다. Pre-charge의 입력이 low 일 때는 두 차동 출력이 모두 high인 상태가 되고, Pre-charge가 high 상태 일때 비교회로에서 저장된 데이터 D와 입력 태그 데이터인 SL이 같을 경우 에만 Match 신호가 high로 인에이블 되어 캠 셀의 완료 신호를 생성한다 [6-8]. 워드라인의 모든 셀의 매치신호가 high로 인에이블 되면 캠에서 히트 상태 임을 캐시 컨트롤러로 알려 주게 된다.

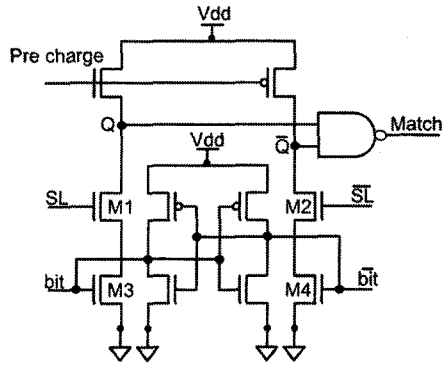


그림 8. 매치에 의한 히트 신호의 발생

Ⅳ. 시뮬레이션 결과

설계된 비동기식 명령어 캐시는 IDEC에서 제공하는 툴을 사용하여 게이트 레벨에서 합성하였으며, [그림 9]와 같이 ESIC 32비트 프로세서와 연동한 환경에서 시뮬레이션 하였다. 플랫폼에서 시뮬레이션 하기위해 CPU와 명령어 캐시 사이에서 핸드쉐이크 프로토콜을 정의하고 래퍼와 라인버퍼를 추가 하였다. 캐시는 프로그램 메모리로부터 블록단위로 데이터를 읽게 설계 되었으나 플랫폼모델에서는 라인 버퍼를 통하여 한 번의 요청신호에 하나의 워드 데이터를 출력 하게 되어 있다. 따라서 라인버퍼는 캐시로부터 한 번의 요청 신호를 받으면 프로그램 메모리로 4번의 데이터를 요청하게 된다. [그림 10]은 시뮬레이션 결과 파형을 보여 준다. 그림에서 캐시요구 신호인 req가 인에이블 된 후 $\Delta t1$ 시간 까지 hit 신호가 high로 천이되지 않기 때문에 프로그램 메모리로 명령어 요청 신호 miss 를 high로 인에이블 시켜 새로운 명령어를 요청 한다. 캐시의 쓰기 신호 we_d를 이용하여 프로그램 메모리의 동작 완료 신호가 high로 천이되는 시점에서 캐시는 새로운 명령어를 저장 한다. 캐시에 새로운 명령어가 저장되면 hit 신호가 high로 인에이블 되고 캐시는 동작완료 신호 ack 에 따라 해당 명령어를 출력한다. 시뮬레이션은 캐시 메모리와 프로세서를 연동 하여 MI벤치마크 프로그램 중 BITC, STRS, QSTR, DHRY, DIJK 총 5개의 프로그램을 실행 하였다. 시뮬레이션에 사용된 프로그램의 경우 같은 연산을 반복적으로 수행하는 연산들이 많고 명령어 캐시만 시뮬레이션 하였기 때문에 99% 이상의 캐시 히트율을 보였다. [표 3]은 각 프로그램의 명령어 캐시의 히트 및 히트미스의 횟수를 나타낸다. 설계된 명령어 캐시는 Best case에서 1.5 ns, Normal case에서 1.572 ns, Worst case에서 2.838 ns의 레이턴시를 갖는다. 시뮬레이션시 프로그램 메모리는 5ns의 레이턴시를 갖는 DRAM을 사용하였다[9]. [표 4]의 결과에 따르면 설계된 명령어 캐시를 사용 할 경우 프로세서가 직접 프로그램 메모리를 엑세스하는 것 보다 68% 이상의 레이턴시 감소 효과를 얻을 수 있다.

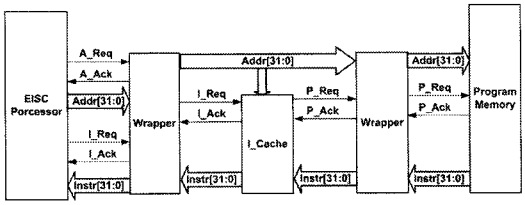


그림 9. EISC 프로세서와 연동 테스트 플랫폼

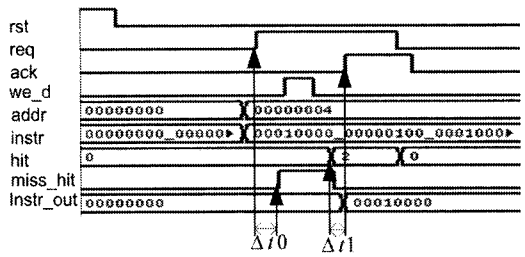


그림 10. 캐시 동작 시뮬레이션 결과 파형

표 3. MI 벤치마크 프로그램 실행 캐시 히트율

MI Bench Program	# of Hit	# of Miss hit	Hit Rate(%)
BITC	723991	89	99
STRS	186040	82	99
QSTR	147966	391	99
DHRY	1150981	268	99
DIJK	5324518	200	99

표 4. 명령어 캐시에 의한 레이턴시 감소율

MI Bench Program	명령어 패치에 의한 레이턴시(ns)		성능 향상율(%)
	without I-Cache	with I-Cache	
BITC	3.62	1.14	68.51
STRS	0.93	0.29	68.82
QSTR	0.74	0.23	71.83
DHRY	5.76	1.81	68.58
DIJK	26.62	8.37	68.56

V. 결론

본 논문에서는 지연무관 모델과 번들지연모델을 적용한 비동기식 명령어 캐시 메모리의 구조를 제안하고 설계하였다. 캐시 메모리는 태그 탐색을 보다 빨리 할

수 있는 캠 및, 4상 연관 맵핑 방법의 구조를 갖는다. 명령어 캐시 데이터 업데이트 정책은 하드웨어 구현시 가장 효율적인 Pseudo-LRU 엔트리 교체 알리즘을 적용하였다.

설계된 비동기 명령어 캐시를 32비트 EISC 프로세서와 연동하는 플랫폼환경을 구축하고 MI 테스트벤치마크 프로그램으로 명령어 캐시의 히트율과 레이턴시 성능을 분석 하였다. 시뮬레이션 결과로 99%의 캐시 히트율을 보여 캐시의 기능적인 동작을 검증하였고, 캐시 히트에 의한 전체의 레이턴시가 기존대비 68%로 감소하는 성능을 얻었다. 설계된 명령어 캐시는 캐시 내부 메모리 램과 캐시의 미스히트 동작을 위하여 번들지연 모델을 동시에 적용하고 있지만 지연소자인 게이트 개수 조정만으로 공정변화에 쉽게 대응할 수 있는 장점이 있다.

참 고 문 헌

[1] J. Montanaro, R. T. Witek, K. Anne, and A. J. Black, "A 160-MHz, 32-b, 0.5-W CMOS RISC microprocessor," IEEE Journal of ISSCC, Vol.31, No.11, pp.1703-1714, 1996(11).

[2] S. N. Kim, S. W. Kim, Y. W. Kim, M. H. Oh, and C. H. Shin, "*Ultra low power asynchronous processor development*," Technical Report 09ZH1230-01-7030P, ETRI, 2009(12).

[3] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. R. Mudge, and B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," Proc. WWC-4.2001, pp.3-14, 2001(12).

[4] D. Hormdee and J. D. Garside, "AMULET3i cache architecture," Proc. ASYNC'2001, pp.152-161, 2001(3).

[5] Z. Wang, S. Das, H. Che, and M. Kumar, "SACCS: Scalable Asynchronous Cache Consistency Scheme for Mobile Environments," Proc. ICDCSW'03, pp.797-802, 2003(5).

[6] J. M. Colmenar, O. Garnica, S. Lopez, J. I. Hidalgo, J. Lanchares, and R. Hermida, "Empirical characterization of the latency of long asynchronous pipelines with data-dependent module delays," Proc. *12th EUROMICROPDP'* 04, pp.311-321, 2004(2).

[7] J. Battogtokh and K. R. Cho, "Design of a DI model-based Content Addressable Memory for Asynchronous Cache," International Journal of Contents, Vol.5, No.2, pp.53-58, 2009(6).

[8] K, Osada, H. Higuchi, K. Ishibashi, N. Hashimoto, and K. Shiozawa, "A 2 ns access, 285 MHz, two-port cache macro using double global bit-line pairs," Proc. ISSCC'97, pp.402-403, 1997(2).

[9] Virantha N. Ekanayak and Rajit Manohar, "Asynchronous DRAM Design and Synthesis," Proc. Asynchronous Circuits and Systems, pp.174-183, 2003(5).

저 자 소 개

전 광 배(Kwang-Bae Jeon) 준회원

- 2007년 8월 : 충북대학교 정보통신(공학사)
- 2007년 9월 ~ 현재 : 충북대학교 정보통신공학과 석사과정

<관심분야> : 비동기 회로, 컴퓨터 구조, 저전력 회로 설계

김 석 만(Seok-Man Kim) 정회원

- 2005년 2월 : 충북대학교 전기전자전공(공학사)
- 2008년 2월 ~ 현재 : 충북대학교 정보통신공학(석사)
- 2008년 3월 ~ 현재 : 충북대학교 정보통신공학과 박사과정

<관심분야> : 고성능 MCU 설계, 저전력 회로 설계

이 제 훈(Je-Hoon Lee) 정회원

- 1999년 2월 : 충북대학교 정보통신공학과(공학사)
- 2001년 2월 : 충북대학교 정보통신공학과(공학석사)
- 2005년 2월 : 충북대학교 정보통신공학과(공학박사)
- 2005년 ~ 2006년 : Univ. of Southern California 방문연구원
- 2006년 ~ 2008년 : 충북대학교 BK21 계약교수
- 2009년 ~ 현재 : 강원대학교 전자과 교수

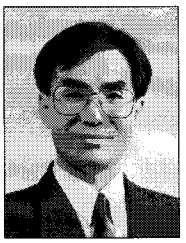
<관심분야> : 고속 마이크로프로세서 설계, 저전력 디자인

오 명 훈(Myeong-Hoon Oh) 정회원

- 1997년 : 전남대학교 컴퓨터공학과(공학사)
- 1999년 : 전남대학교 컴퓨터공학과(공학석사)
- 2001년 ~ 2002년 : University of Manchester 방문 연구원
- 2005년 : 광주과학기술원 정보통신공학과(공학박사)
- 2005년 ~ 현재 : 한국전자통신연구원 서버플랫폼 연구팀 선임연구원

<관심분야> : 동기회로 설계, GALS 시스템 설계, 저전력 회로 설계

조 경 록(Kyoung-Rok Cho) 정회원



- 1977년 : 경북대학교 전자공학과(공학사)
- 1989년 : 일본 동경대학교 전자공학과(공학석사)
- 1992년 : 일본 동경대학교 전자공학과(공학박사)
- 1979년 ~ 1986년 : (주)금성사 TV연구소 선임연구원
- 1999, 2006년 : Oregon State University 객원교수
- 1992년 ~ 현재 : 충북대학교 전자정보대학 교수

<관심분야> : 통신시스템 LSI설계, 저전력 고속회로 설계, Platform 기반의 SoC 설계