Write Back 모드용 FIFO 버퍼 기능을 갖는 비동기식 데이터 캐시

Design of an Asynchronous Data Cache with FIFO Buffer for Write Back Mode

박종민*, 김석만*, 오명훈**, 조경록*
충북대학교 정보통신 공학과*, 한국전자통신연구원**

Jong-Min Park(jmpark@hbt.cbnu.ac.kr)*, Seok-Man Kim(smkim@hbt.cbnu.ac.kr)*,
Myeong-Hoon Oh(mhoonoh@etri.re.kr)**, Kyoung-Rok Cho(krcho@chungbuk.ac.kr)*

요약

본 논문에서는 32bit 비동기 임베디드 프로세서용 쓰기 버퍼 기능을 갖는 데이터 캐시 구조를 제안하고 성능을 검증하였다. 데이터 캐시는 비동기 시스템에서 메인 메모리 장치와 프로세서 사이의 데이터 처리 속도 향상을 목적으로 한다. 제안된 데이터 캐시의 메모리 크기는 8KB, 매핑 방식으로는 4 words(16byte)의 라인 크기를 가지며, 사상 기법으로는 4 way set associative, 교체 알고리즘으로는 pusedo LRU방식을 사용하였으며, 쓰기 정책을 위한 dirty 레지스터와 쓰기 버퍼를 적용시켰다. 설계한 데이터 캐시는 0.13-㎛ CMOS공정으로 합성하였으며, MI벤치마크 검증 결과 평균 히트율은 94%이고 처리 속도가 46% 향상되었다.

■ 중심어 : | 데이터 캐시 | 쓰기 버퍼 | 비동기 FIFO | 비동기 시스템 |

Abstract

In this paper, we propose the data cache architecture with a write buffer for a 32bit asynchronous embedded processor. The data cache consists of CAM and data memory. It accelerates data up lood cycle between the processor and the main memory that improves processor performance. The proposed data cache has 8 KB cache memory. The cache uses the 4-way set associative mapping with line size of 4 words (16 bytes) and pseudo LRU replacement algorithm for data replacement in the memory. Dirty register and write buffer is used for write policy of the cache. The designed data cache is synthesized to a gate level design using 0.13-㎛ process. Its average hit rate is 94%. And the system performance has been improved by 46.53%. The proposed data cache with write buffer is very suitable for a 32-bit asynchronous processor.

■ keyword : | Data Cache | Write Buffer | Asynchronous FIFO | Asynchronous System |

I. 서 론

현재 임베디드 프로세서는 소형 및 휴대용기기에 많이 사용되고 있으며, 다양하면서 복잡해진 소프트웨어를 지원하기 위해 고속화되어가고 있다. 이에 따라 임베디드 프로세서 내에 캐시를 구성하여 데이터 처리 성

* 이 논문은 2008년도 충북대학교 학술연구지원사업의 연구비 지원에 의하여 연구되었음
접수번호 : #100531-003
접수일자 : 2010년 05월 31일
심사완료일 : 2010년 06월 18일
교신저자 : 박종민, e-mail : jmpark@hbt.cbnu.ac.kr

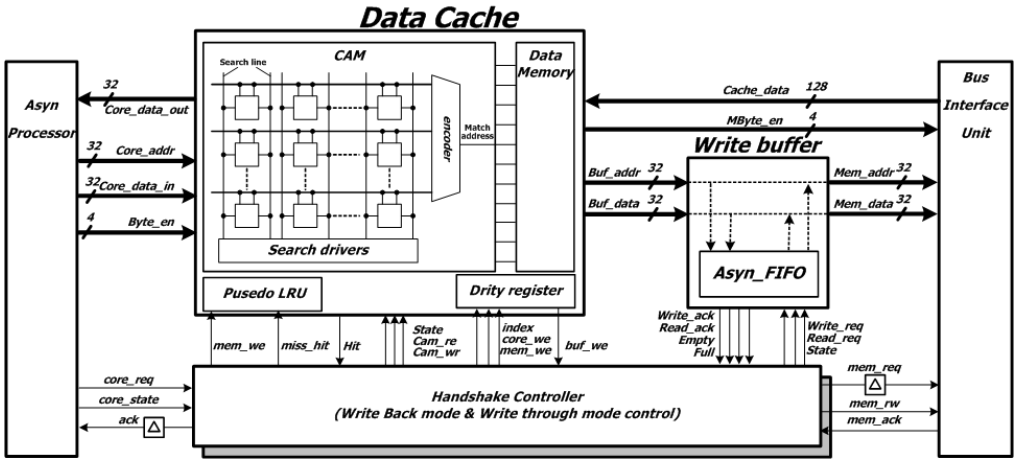


그림 1. 데이터 캐시 전체 구성도

능을 높이는 방법이 적용되고 있다. 일반적으로 캐시는 SRAM으로 설계되며 프로세서의 요청에 고속의 데이터 탐색 및 전달을 목적으로 설계되어 진다. 캐시의 크기를 크게 늘리면 효율성이 높일 수 있다. 그러나 빈번한 접근으로 인한 전력 손실이 커진다. 비동기 설계 기법은 클럭을 사용하지 않아 전력 손실을 줄일 수 있는 기법 중에 하나이다.

비동기 시스템은 신호 및 논리 처리 기준 시간을 정의하는 클럭을 사용하지 않고 회로를 설계하는 방식이다. 인접한 두 기억 소자 간에 미리 지정된 데이터 전달 방식(handshaking protocol)을 사용하여 시스템의 상태를 입력된 값이나 내부의 동작으로 표시하고 어떤 사건이 발생된 후 또는 다른 사건까지 신호의 값을 유지하는 특징을 가짐으로써, 동작이P 필요 없는 모듈들이 휴지 상태에 있을 때 전력을 소모하지 않고 발열 및 노이즈가 없는 것을 특징으로 하는 설계 방식이다. 본 논문에서 데이터 캐시의 고속화와 저전력화을 위해 쓰기 버퍼 기능을 갖는 비동기 데이터 캐시를 제안한다. 제안하는 비동기 데이터 캐시의 구조는 데이터 캐시, 쓰기 버퍼[2]와 핸드쉐이크 컨트롤러로 구성되어 있다. 그리고 write through모드와 write back모드 중 선택적으로 동작하는 비동기식 쓰기 버퍼의 구조도 제안한다. 제안된 캐시는 0.13㎛ 라이브러리를 사용하여 합성한 후 비동기식 EISC 프로세서[3]와 연동하여 MI 벤치 마크[4] 프로그램을 실행시켜 성능을 평가하였다.

본 논문의 구성은 다음과 같다. 2장에서는 비동기식 데이터 캐시의 전체 구조와 적용된 회로 기술을 소개하고, 핸드쉐이크 컨트롤러를 설명하며, 그리고 쓰기 정책을 기술한다. 3장에서는 비동기 FIFO(first in first out) 방식을 이용한 쓰기 버퍼를 기술하고 4장에서는 시뮬레이션 결과를 언급하고, 5장에서는 논문의 결론에 대해 기술한다.

II. 비동기 데이터 캐시 설계 및 구현

본 논문에서 구현한 비동기식 데이터 캐시는 32bit 임베디드 프로세서용으로 설계하였다. [그림 1]은 설계된 데이터 캐시의 전체 구성도이다. 제안된 데이터 캐시의 구조는 태그 메모리와 데이터 메모리를 포함하고 있는 캐시와 쓰기 버퍼 및 핸드쉐이크 컨트롤러로 구성된다. 데이터 캐시는 메인 메모리 접근 지연시간(memory access latency)을 줄이기 위해 사용되나 프로세서의 모든 데이터를 포함 할 수 없기 때문에 공간 지역성과 시간 지역성을 활용하여 가장 최근에 사용된 데이터를 저장하게 된다. 그리고 충돌 캐시 미스(conflict cache miss)를 줄이는데 가장 효율적인 세트 연관 매핑 방식(set associative mapping)을 사용하였다. 주소 태그(tag)를 저장하기 위하여 램이 아닌 내용 연관 기억 장

치인 CAM(contents addressable memory)으로 구현하였다. 매핑 방식으로는 4상 연관 매핑 구조로 설계하였다. 그리고 pesudo LRU 데이터 교체 방식을 따르며, 쓰기 정책으로는 write through와 write back 방식을 적용하였으며, write back 모드를 위한 dirty 레지스터와 쓰기 버퍼를 적용시켰다[5][6].

2.1 지연모델

본 논문에서 제안한 비동기식 데이터 캐시는 DI(delay-insensitive) 모델과 BD(bundled- delay)모델을 적용[7]하여 설계하였다. 캐시는 프로세서가 데이터 요청시 주소 정보가 태그 메모리에 저장된 태그와 비교하여 일치(*hit*)하면, 매핑된 주소의 데이터를 전달한다. 태그 메모리는 CAM으로 설계되며, 주소 정보의 일부를 저장된 태그와 비교하여, 동일하면 match 신호를 발생한다. 이것은 특정한 지연 시간이 존재하지 않고 회로 동작에 의해 신호가 발생되는 DI 모델이 사용된다. DI 지연모델을 동작시키기 위해 회로를 CAM으로 설계하였다. 그러나 프로세서로부터 요청된 주소 중 태그가 CAM에 없을 경우 캐시는 미스 히트 상황이라고 판단하고, match신호를 발생하지 않는다. 일정 지연 시간이 지나도 match 신호가 없으면 메인 메모리에 데이터 요청신호를 발생시켜준다. 이때 사용된 정해진 지연 시간을 bundled 지연 시간이라고 한다. 메모리를 엑세스 할때는 데이터의 출력을 확인할 수 있는 신호생성이 불가 하므로 일정시간 후에 출력이 얻어지는 것으로 판정하는 BD지연으로 신호를 생성하였다.

2.2 데이터 캐시

제안된 데이터 캐시는 4웨이 세트 연관 사상 방식을 사용하여 구현하였으며, 교체 정책(replacement policy)으로는 pseudo LRU(least recently used) 알고리즘을 사용하였다.

캐시는 [그림 2]와 같이 태그 메모리와 데이터 메모리로 구성되었다. 태그 메모리는 CAM으로 설계되었으며, 태그는 21bit로 프로세서에서 요청한 데이터가 캐시에 저장되어 있는지 판단하고 인덱스는 요청된 데이터가 있는 블록의 위치를 알려준다. [그림 3]은 데이터 캐시내의 태그 메모리와 데이터 메모리간의 블록 다이어그램이다. 데이터 주소가 태그 메모리를 거쳐 데이터 메모리를 엑세스하고, 데이터는 쓰기 버퍼를 통해서 외부 메인 메모리를 엑세스한다.

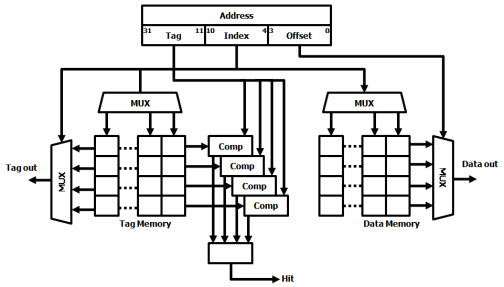


그림 2. 데이터 캐시 구조

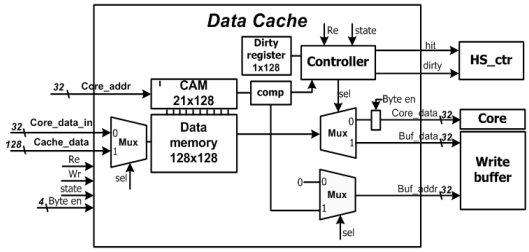


그림 3. 데이터 캐시 블록 다이어그램

2.3 핸드쉐이크 컨트롤러

핸드쉐이크 컨트롤러는 프로세서로부터 캐시 제어 신호를 받고 이에 대한 결과를 프로세서로 전달하는 역할을 한다. 쓰기 정책에 의해 비동기 쓰기 버퍼를 컨트롤을 하게 된다. [그림 4]는 캐시의 읽기 접근 타이밍도이다. 프로세서에서 주소(*core_addr*)와 함께 데이터 읽기 요청신호(*cam_re*)를 전달받으면 태그 메모리는 저장된 주소와 비교한다. 같은 주소를 찾으면 캐시내 메모리의 데이터를 프로세서에 전달하고 응답 신호(*ack*)를 전송한다. 데이터 요청에 대한 캐시 읽기 동작을 완료하게 된다. 또한 프로세서에서 데이터 캐시내 메모리에 연산 결과를 갱신해야 한다. 프로세서에서 쓰기 요청신호(*core_we*)가 오면, 주소와 일치하는 데이터 메모리에 연산 결과를 저장하게 된다. 그리고 프로세서로 쓰기 응답 신호(*ack*)를 보내고 동작을 완료한다.

[그림 5]는 프로세서에서 요청한 주소의 데이터가 캐

시에 없는경우 읽기 접근 실패 타이밍에 대해 나타내었다. 데이터 요청 신호가 전달되고, 일정 시간이내에 *hit* 신호가 발생되지 않으면 캐시 내에 유효 데이터가 없다고 판단하고 핸드쉐이크 컨트롤러는 메인 메모리에 저장된 데이터를 요청(*mem_req*)한다. 데이터 캐시는 메인 메모리의 데이터(*cache_data*)와 요청한 주소(*core_addr*)를 캐시에 교체하고 *byte_en*의 상태 값에 따라 프로세서로 데이터(*core_data_out*)를 전달하게 된다.

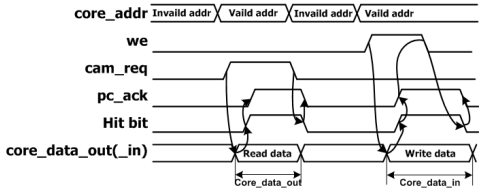


그림 4. 데이터 캐시 읽기 접근 타이밍도

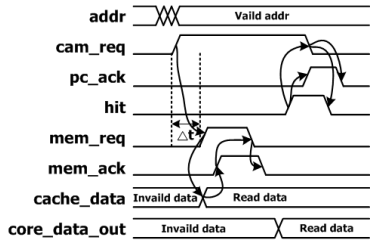


그림 5. 데이터 캐시 읽기 접근 실패 타이밍도

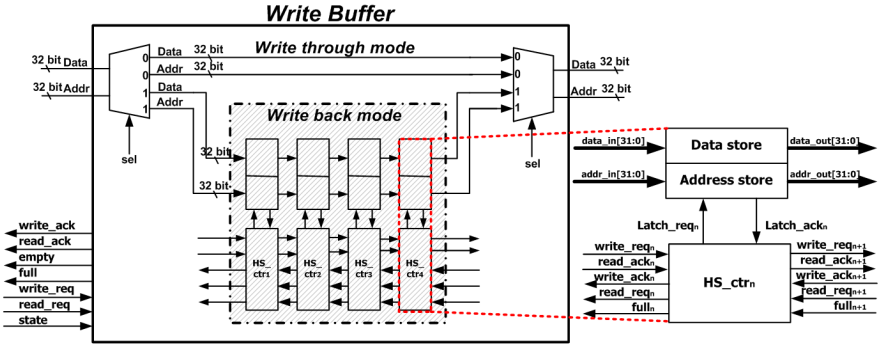
다음은 핸드쉐이크 컨트롤 블록의 입출력 신호에 대한 설명이다. *re*는 데이터 요청 신호이고, *wr*는 데이터 캐시에 쓰기 신호이다. *ack*는 데이터 캐시 동작이 완료되었다는 응답신호이다. *hit*는 프로세서로부터 요청된 주소의 데이터의 유무 상태를 확인하는 신호이다. *state*는 'low'이면 write through 모드, 'high'이면 write back모드로 동작한다. *cam_re*는 데이터 읽기 신호이다. *cam_wr*는 주소와 데이터를 쓰기 신호이다. *dirty*는 write back모드 경우 데이터 캐시의 데이터 메모리 정보가 갱신되었는지 확인하는 신호이다. *mem_req*는 데이터 캐시에서 미스 발생시 메인 메모리의 데이터를 요청하는 신호이다. 신호 *mem_ack*는 데이터 캐시에서 요청된 신호를 처리 했다는 응답 신호이다.

2.4 쓰기 정책

캐시는 메인 메모리의 데이터 일부를 임시 저장하고 있다. 쓰기 동작에서의 데이터 일관성을 유지하기 위해 모든 쓰기 동작을 캐시로만 하는 write back 모드와 모든 쓰기를 데이터 캐시와 외부 메모리에 동시에 수행하는 write through 모드 중 한가지로 동작한다. Write through 모드는 프로세서로부터 갱신된 데이터를 메인 메모리에 직접 저장하는 방식으로 구조가 간단하다. 그러나 write through 모드는 매 쓰기 동작마다 메인 메모리에 접근해야 하므로 스토어 명령어 동작이 완료될 때까지 레이턴시가 길다는 단점이 있다. 이와 같은 단점을 보완하기 위해 본 논문에서는 write back 모드를 병행으로 적용하였다. 블록이 교체되기 전에 내용이 캐시 내에서 변경되었으면서도 메인 메모리에 갱신되지 않았는지 반드시 확인하는 과정을 거친다. 즉 write back 모드로 동작 시에 데이터가 갱신이 되면 dirty 레지스터에 플래그하여 데이터 교체될 때 dirty 레지스터의 상태가 'high'면 메인 메모리에 데이터를 갱신시킨다.

III. 비동기 쓰기 버퍼

비동기 쓰기 버퍼는 메인 메모리에 쓰기 동작을 위한 임시버퍼로 사용된다. 본 논문에서 제안하는 비동기식 쓰기 버퍼는 메인 메모리에 스토어 동작 시 구동되며, 구성은 FIFO구조로 설계하였고 4개의 엔트리로 구성되었다. 각각의 엔트리는 4바이트 주소 정보와 4바이트 데이터 정보를 저장할 수 있는 래치와 이를 제어하기 위한 핸드쉐이크 블록으로 구현되었다. [그림 6]은 write through와 back모드로 동작할 수 있는 비동기 쓰기 버퍼의 구조이다. Write through모드는 입력된 데이터와 주소를 메인 메모리에 직접 전달된다. Write back 모드는 입력된 데이터와 주소를 래치로 설계된 store에 저장하게 된다. Store에 저장된 데이터는 메인 메모리가 사용 가능한 상태일 때 독립적으로 메인 메모리에



(a) 쓰기 버퍼 전체 구성도 (b) FIFO 블록 다이어그램

그림 6. 쓰기 버퍼 구성도

저장된다. [그림 6](a)는 쓰기 버퍼의 구성도이며, sel 신호에 따라 write through/back모드로 동작한다. [그림 6](b)는 엔트리 하나에 대한 입출력 신호에 대해 보여주는 비동기 FIFO 블록 다이어그램이다. 래치를 제어하기 위해 엔트리마다 핸드쉐이크 컨트롤 블록을 구현하였다.

쓰기 버퍼는 엔트리의 상태를 외부 핸드쉐이크 컨트롤러에 알려줘야 한다. 마지막 엔트리까지 데이터와 주소가 저장되었으면 *full*신호를 외부 핸드쉐이크 컨트롤러에 전달한다. 데이터 캐시에 *full*신호가 전달되면 대기 상태 가 되고 쓰기 버퍼에 *full*신호가 해제될 때 까지 대기 상태가 된다. 메인 메모리에 데이터와 주소를 전달하면 *full*신호는 해제된다. 또한 엔트리의 모든 데이터와 주소를 메인 메모리 전달하면 *empty* 신호를 외부 핸드쉐이크 컨트롤러에 전달된다.

[그림 7]은 write back 모드 동작 시 쓰기 동작에 대한 엔트리 타이밍도이다. 쓰기 버퍼는 write back모드로 동작하고 있으며, 캐시에서 *write_req*신호를 'high'로 보낸다. *latch_ack*의 상태가 'high'이면 래치의 입력단과 출력단의 주소와 데이터가 같고, *latch_ack*의 상태가 'low'이면 다르다. *Hs_ctr*는 래치에 저장된 데이터와 다른 데이터가 입력 단에 있는 상태에서 *write_req*가 'high'면 래치에 저장 요청신호를 보낸다. 저장이 완료되면 응답신호를 앞단의 엔트리로 전송하게 되면 엔트리의 쓰기 동작이 완료된다. 이 후 다음 엔트리에서 들어오는 *full*신호 상태를 확인하여, 'high'이면 데이터가 저장되어 있는 상태이고, 'low'이면 엔트리는 쓰기 가능한 상태이다.

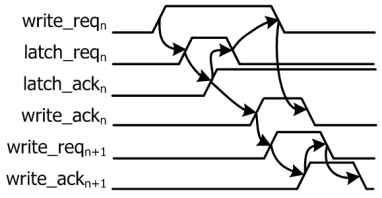


그림 7. 쓰기 동작 타이밍

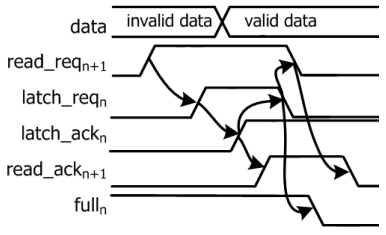


그림 8. 읽기 동작 타이밍

[그림 8]은 write back 모드 동작 시 읽기 동작에 대한 엔트리 타이밍도를 보여주고 있으며, 읽기 동작은 쓰기 버퍼에 저장되어 있던 데이터가 메인 메모리에 데이터를 전달해주고 쓰기 버퍼의 다른 엔트리의 유효 데이터를 저장하는 동작에 대한 것이다. 신호 $read_req_{n+1}$에 따라 읽기 동작이 실행되며, 읽기 요청 신호를 받은 엔트리의 데이터를 다음 엔트리에 전달시켜야 한다. 이를 위해 읽기 요청신호가 오면 $latch_ack_n$신호 상태를

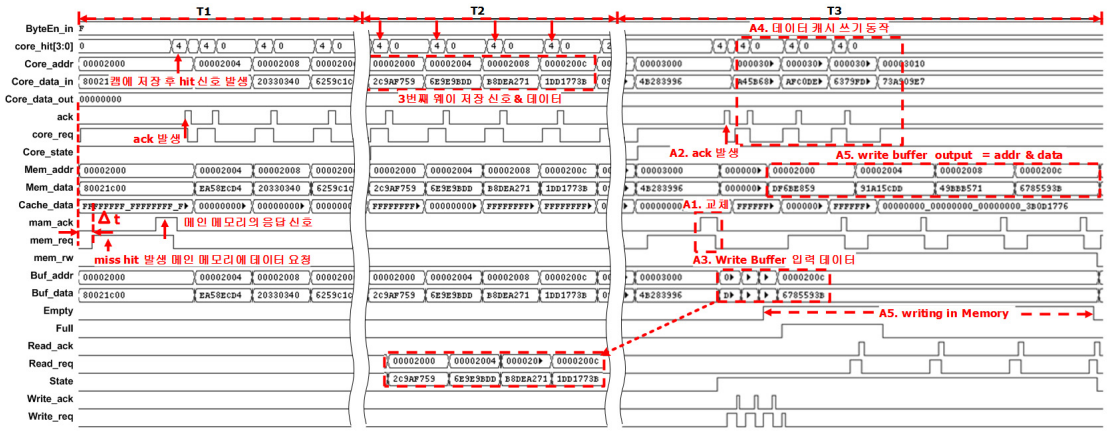


그림 10. Write Back 시뮬레이션

확인하여 'low'면 데이터 저장(*latch_req*$_n$)한다. 이 후 *latch_ack*$_n$신호가 상승 클럭 발생하면 데이터 저장이 완료되고, 읽기 응답신호(*read_ack*)를 "high"로 천이 시킨다. 동시에 *latch_req*가 "low"가 되면서 *full*신호를 "low"상태로 천이시키고 읽기 요청신호가 "low"로 천이되면서 응답신호도 "low"로 천이되면서 읽기 동작이 완료된다.

VI. 시뮬레이션 결과

설계된 비동기식 데이터 캐시는 IDEC에서 제공하는 툴을 사용하여 게이트 레벨로 합성하였으며, ESIC 32비트 비동기 프로세서와 연동하여 시뮬레이션을 하였다.

4.1 테스트 플랫폼

프로세서와 연동 테스트를 하기위해 입출력 신호와 핸드쉐이크 프로토콜을 정의하였다. 플랫폼에서 시뮬레이션 하기 위해 EISC 프로세서와 데이터 캐시 사이에 래퍼를 설치하고 캐시와 프로그램 메모리 사이에 라인 버퍼를 설치하였다. 데이터 캐시의 메모리 교체는 블록 단위로 전송이다. 그러나 라인버퍼와 프로그램 메모리 간에는 32bit단위로 데이터 교환이 된다. 캐시의 데이터 요청이 발생하면 라인 버퍼와 프로그램 메모리는 4번의 핸드쉐이크가 발생된다.

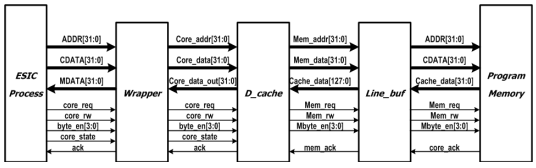


그림 9. EISC 프로세서와 연동 테스트 플랫폼

4.2 벤치 마크 테스트

[그림 10]의 시뮬레이션은 데이터 캐시가 쓰기 모드로 동작하고 있다. 데이터 저장 요청 신호를 보냈으나 Δt 시간까지 hit신호가 "high"로 천이되지 않자 미스 히트 신호가 발생되었다. 메인 메모리에 데이터 요청신호를 보낸다. 메인 메모리의 응답 신호가 발생되면 주소와 데이터를 캐시 내에 저장하고 완료 신호를 전달한다. 이후 같은 태그 값을 가진 주소로 인해 히트 상태가 된다. 그러나 *T3*구간에서 저장된 태그와 다른 태그를 가진 데이터를 저장해야 한다. 그래서 *T1, T2*에서 교체된 데이터는 데이터의 일관성을 위해 메인 메모리에 전달해야한다. *A1*에서 캐시 내용이 교체되고 새로운 데이터가 저장 완료 신호가 발생(*A2*)되면서 쓰기 버퍼로 데이터가 전달(*A3*)된다. 프로세서는 새로운 쓰기 동작(*A4*)을 실행시키고, 쓰기 버퍼에 있는 데이터는 핸드쉐이크 컨트롤러의 제어 신호에 의해 메인 메모리로 전송된다.

MI벤치 마크 프로그램인 Qsort, String Search,

Dhrystone, Bitcount등 총 4개의 프로그램으로 테스트 하였다. 5ns의 액세스 시간을 갖는 프로그램 메모리[8]와 라인버퍼 지연을 5ns라고 설정하여 측정하였다. [표 1]의 결과에서는 비동기식 데이터 캐시의 평균 히트율은 94%정도이며 시뮬레이션 결과에 따르면 데이터 캐시의 접근 시간은 5.35ns를 갖는다. [표 2]의 결과는 설계한 데이터 캐시를 사용하면 프로세서가 메인 메모리를 직접 액세스(without D-cache)하는 것보다 46%정도 시스템 향상 시킬 수 있다.

표 1. 데이터 캐시 히트율

Program	hit ratio (%)	Data cache(갯수)	
		hit	miss
Qsort	85	68170	11253
String Search	95	24564	1031
Dhrystone	98	446684	7145
Bitcount	98	99551	1240

표 2. 데이터 캐시에 의한 실행 시간 감소율

Program	실행시간 감소율(%)	Run time(ms)	
		without D-cache	with D-cache
Qsort	46.57	10.55	5.63
String Search	46.52	10.76	5.75
Dhrystone	46.51	83.04	44.42
Bitcount	46.51	48.22	25.79

V. 결론

본 논문은 비동기식 프로세서를 위한 쓰기 버퍼 기능을 갖는 비동기 데이터 캐시 구조를 제안하고 설계하였다. 제안하는 데이터 캐시는 캠을 이용하여 태그 탐색을 빠르게 하고 4상 연관 매핑 구조로 설계하였다. 그리고 pesudo LRU 데이터 교체 방식을 따르며, 쓰기 정책으로는 write through와 write back 방식을 적용하였으며, write back 모드를 위한 dirty 레지스터와 쓰기 버퍼를 적용시켰다. 설계된 데이터 캐시 구조는 핸드쉐이크 컨트롤러간의 데이터 표현 방식에 있어서 4상 방식을 사용하였다.

설계된 비동기 데이터 캐시를 32비트 EISC 프로세서와 연동하는 플랫폼 환경에서 벤치 마킹 프로그램을 사용하여 히트율과 레이턴시 성능을 분석하였다. 데이터 캐시의 평균 히트율은 94%이며, 데이터 캐시가 적용되지 않는 시스템 보다 처리 속도가 46%향상되었다.

참 고 문 헌

[1] J. Pangjun and S. Sapatnekar, "Low-power Clock Distribution Using Multiple Voltages and Reduced Swings," IEEE Journal of TVLSI, Vol.10, No.2, pp.309-319, 2002(6).

[2] X. Wang, T. Ahonen, and J. Nurmi, "A Synthesizable RTL Design of Asynchronous FIFO," Proc. ISSOC'2004, pp.123-128, 2004(11).

[3] 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).

[4] D. Hormdee and J. D. 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).

[5] D. Hormdee and J. D. Garside, "AMULET3i cache architecture," Proc. ASYNC'2001, pp.152-161, 2001(5).

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

[7] 전광배, 김석만, 이제훈, 오명훈, 조경록, "혼합 지연 모델에 기반한 비동기 명령어 캐시 설계", 한국콘텐츠학회논문지, 제10권, 제3호, pp.64-71, 2010(3).

[8] 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

박 종 민(Jong-Min Park) 준회원

- 2006년 2월 : 서원대학교 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

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
- 1999년, 2006년 : Oregon State University 객원교수
- 1992년 ~ 현재 충북대학교 전자정보대학 교수

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