논문 2009-46SD-1-9

OpenRISC 코어의 성능향상을 위한 캐쉬 구조 설계

(Cache Architecture Design for the Performance Improvement of OpenRISC Core)

정 홍 균*, 류 광 기**

(Hongkyun Jung and Kwangki Ryoo)

요 약

최근 마이크로프로세서의 성능이 빠르게 향상됨에 따라 주 메모리의 접근 시간이 증가하고 있어 캐쉬의 필요성이 증대되고 있다. 직접사상 캐쉬는 주 메모리의 각각의 블록이 하나의 캐쉬 라인에 사상되는 구조로서 사상되는 규칙이 간단하지만 서로 다른 블록이 하나의 캐쉬 라인에 사상될 경우 블록의 충돌에 의한 접근 실패율이 집합연관 캐쉬에 비해 높아진다. 본 논문에서는 OpenRISC 코어의 직접사상의 단점을 개선하기 위해 사원 집합연관 캐쉬 구조를 제시한다. 제시한 캐쉬는 주 메모리의 네 개의 블록이 하나의 캐쉬 라인에 사상되는 구조로서 직접사상 캐쉬에 비해 접근 실패율이 감소한다. 또한 라인 교체 방식으로 Pseudo-LRU 방식을 채택하여 LRU 정보를 저장하는 비트 수를 감소시켰다. FPGA 에뮬레이션을 이용하여 사원 집합연관 캐쉬를 포함한 OpenRISC 코어를 검증하였고, 테스트 프로그램을 이용하여 성능을 측정한 결과, 사원 집합연관 캐쉬를 포함한 OpenRISC 코어의 성능이 기존의 OpenRISC 코어의 성능에 비해 약 50% 향상되었고, 미스율은 15%이상 감소하였다.

Abstract

As the recent performance of microprocessor is improving quickly, the necessity of cache is growing because of the increase of the access time of main memory. Every block of direct-mapped cache maps to one cache line. Although the mapping rule is simple, if different blocks map to one cache line, the miss ratio will be higher than the set-associative cache due to conflicts. In this paper, for the improvement of the direct-mapped cache of OpenRISC, 4-way set-associative cache is proposed. Four blocks of the main memory of the proposed cache map to one cache line so that the miss ratio is less than the direct-mapped cache. Pseudo-LRU Policy, which is one of the Line Replacement Policies, is used for decreasing the number of bits that store LRU value. The OpenRISC core including the 4-way set-associative cache was verified with FPGA emulation. As the result of performance measurement using test program, the performance of the OpenRISC core including the 4-way set-associative cache is higher than the previous one by 50% and the decrease of miss ratio is more than 15%.

Keywords : Cache, set-associative, OpenRISC

I. 서 론

캐쉬 메모리 시스템은 데이터의 지역특성을 이용하여 주 메모리보다 작고, 빠른 속도를 갖는 캐쉬 메모리에 빈번하게 사용되는 데이터를 저장해두고 억세스 함으로써 프로세서와 메모리의 속도차이로 인한 데이터 억세스 지연시간을 감소시키는 효과적인 방법이다[1].

최근 프로세서의 속도 개선은 급격히 향상된 반면 메모리의 속도 개선은 더디게 이루어져 상대적으로 프로세서가 주 메모리를 접근하는데 소요되는 시간이 증가했다. 따라서 프로세서의 주 메모리 접근을 최대한 줄이는 캐쉬 메모리의 개발이 필수적이다[2~3].

본 논문에서는 OpenRISC 코어에 구현되어 있는 직접사상 캐쉬의 높은 접근 실패율을 개선하기 위해 사원

* 학생회원, ** 평생회원, 한밭대학교 정보통신전문대학원
(Graduate School of Information and Communication, Hanbat National University)

※ 본 연구는 IDEC의 지원 및 지식경제부 출연금으로 ETRI, 시스템반도체산업진흥센터에서 수행한 IT SoC 핵심설계인력양성사업의 연구결과임.
접수일자: 2008년11월3일, 수정완료일: 2009년1월5일

집합연관 캐쉬 구조를 제시하였다. 사원 집합연관 캐쉬는 네 개의 메모리 블록을 한 개의 캐쉬 블록에 사상되는 구조로 되어있어 직접사상 캐쉬에 비해 블록 충돌에 의한 접근 실패율이 낮다. 제안된 캐쉬의 교체 알고리즘으로는 Pseudo-LRU 알고리즘을 사용하였다.

Ⅱ. 관련 연구

1. 캐쉬의 집합연관성

주 메모리의 일부분을 캐쉬에 저장하기 위해서는 용량이 큰 주 메모리의 주소 공간을 작은 캐쉬의 주소 공간에 매핑하기 위한 규칙이 존재하며 이들은 가속화를 위해 하드웨어로 구현된다. 캐쉬의 주소공간과 주 메모리의 주소공간 매핑 관계에 따라 캐쉬는 직접 사상 방식, 집합연관 방식, 완전 연관 방식으로 분류할 수 있다. 직접사상 캐쉬는 주 메모리상의 임의의 한 라인에 대해 캐쉬 내에 고정된 하나의 위치만을 제공함으로써 빠른 액세스 속도와 손쉬운 구현을 제공한다. 직접 사상 캐쉬는 메모리 참조가 발생하면 메모리 주소의 캐쉬 라인 인덱스 부분을 이용하여 캐쉬 내의 태그와 라인 데이터를 읽어 낸다[4~5]. 논리적으로 가장 이상적인 캐쉬의 형태가 바로 완전 연관 매핑을 갖는 캐쉬이며 완전 연관 캐쉬라 부른다. 완전 연관 캐쉬에서 데이터가 들어갈 수 있는 주 메모리의 위치와 캐쉬 메모리 상의 위치는 아무런 상관 관계를 갖지 않는다. 완전연관 캐쉬는 태그 검색과 라인 데이터 읽기를 동시에 수행하는 이후에 액세스하는데 상대적으로 많은 시간이 소요된다. 또한 캐쉬 미스시 라인 치환의 대상이 캐쉬 내의 모든 라인 엔트리가 되므로 그 수가 너무 많아 치환 알고리즘을 하드웨어로 구현시 복잡도가 매우 크다[4, 6].

완전연관 방법과 직접사상 방법을 상호 보완하여 주 메모리의 라인이 캐쉬의 특정 라인 집합에 매핑될 수 있도록 한 방법이 집합 연관 캐쉬다. 집합 연관 캐쉬는 주 메모리의 한 라인이 매핑될 수 있는 영역을 하나의 라인이 아닌 몇 개의 라인으로 구성된 집합으로 넓힘으로써 충돌에 의한 캐쉬 미스의 증가를 막는다. 또한 집합 내의 제한된 수의 라인 엔트리만을 비교 대상으로 하므로 직접 사상 캐쉬에 견줄 수 있을 정도의 처리 속도를 낼 수 있다[5~7].

2. OpenRISC 코어의 직접사상캐쉬

OpenRISC 코어의 직접사상 캐쉬는 크기가 8KB이고

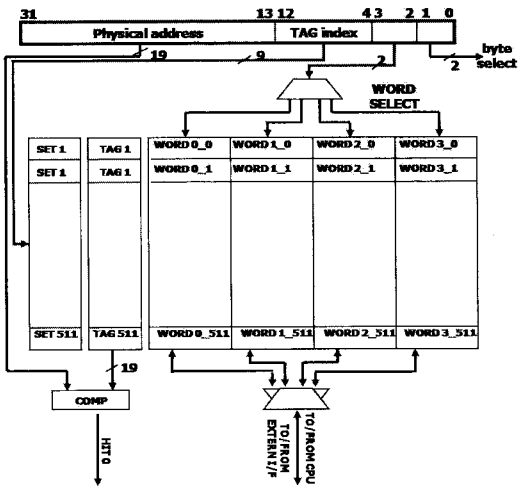

그림 1. OpenRISC 코어의 캐쉬 구조
Fig. 1. Cache architecture of OpenRISC core.

주소태그 램, 네 개의 워드를 저장하는 데이터 램, 주소 태그와 물리 주소를 비교하는 태그 비교기, 워드 선택기로 구성되어 있다[8]. 그림 1은 OpenRISC 코어의 캐쉬 구조이다. 기존의 OpenRISC 프로세서의 캐쉬는 집합연관 캐쉬의 각 집합의 태그 비교, 각 집합의 데이터 멀티플렉싱 회로가 포함되어 있지 않아 간단하고 빠르다는 장점이 있지만, 주 메모리의 주소에 따라 매핑되는 캐쉬 라인이 하나밖에 존재하지 않기 때문에 충돌에 의한 캐쉬 미스로 인해 성능이 많이 떨어진다.

Ⅲ. OpenRISC 코어

본 논문에서 사용된 OpenRISC 코어는 Opencores에서 제공하는 OpenRISC 1000 구조를 갖는 RISC 코어이다. OpenRISC 코어는 누구나 쉽게 재사용 할 수 있고 수정 가능한 LGPL 기반으로 라이센스 되어 있다. 또한 합성 가능한 Verilog HDL로 기술되어 있기 때문에 특정 공정을 사용하여 SoC 및 어플리케이션 시스템을 개발하고자 하는 곳에서 설계 조건에 맞게 칩 제작이 가능하다. OpenRISC 코어는 명령어/데이터 버스 및 메모리가 분리된 하버드 구조로 된 MIPS 기반 RISC 코어이다. RISC 코어의 특징에 맞게 5단 파이프라인 구조를 채택 하였으며, 임베디드 시스템을 타깃으로 실시간 운영체제 지원을 위한 메모리 관리 장치를 지원하고 프로세서 내에 곱셈 및 누산기 유닛을 통해 기본적인 DSP 기능을 지원하고 있다. 또한 코어 내부의 온칩 메모리를 제외한 로직들의 게이트 수가 비교적 작고, 저 전력을 위한 파워 관리 블록과 외부 시스템과의 인터페이스

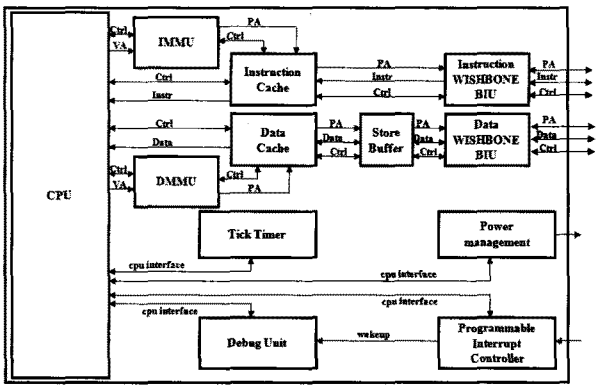

그림 2. OpenRISC 코어 구조
Fig. 2. The architecture of OpenRISC core.

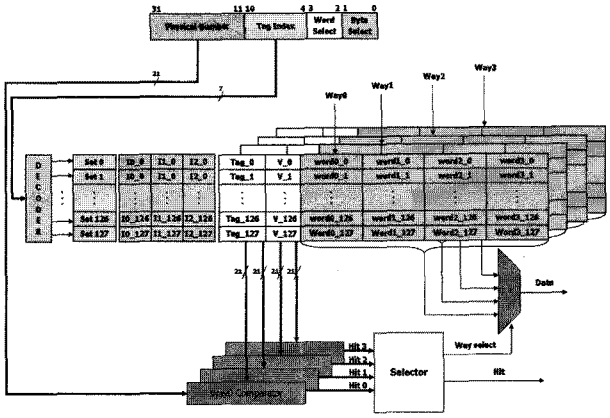

그림 3. 사원 집합연관 캐쉬의 구조
Fig. 3. The architecture of proposed cache.

를 통한 쉽고 빠른 디버깅 환경, 프로그램 가능한 인터럽트 인식 및 처리, WISHBONE 표준 인터페이스를 통해 명령어 및 데이터 인터페이스를 구성하여 쉽게 IP들의 추가 및 수정이 가능하다는 점에서 포터블, 임베디드 시스템에 적합하고 적은 수의 게이트와 공개된 RTL로 인해 저비용, 저 전력의 시스템에 사용 될 수 있다[8~9]. 그림 2는 OpenRISC 코어의 내부 구조이다.

IV. 사원 집합연관 캐쉬

1. 사원 집합연관 캐쉬 원리

본 논문에서는 RISC 코어에서 사용했던 기존의 직접 사상 방식의 단점, 즉 한 개의 캐쉬 블록에 여러 개의 메모리 블록이 사상될 때의 높은 접근 실패율을 보완하기 위하여 사원 집합 연관 캐쉬를 설계하였다. 사원 집합 연관 캐쉬는 사실상 병렬적으로 동작하는 네 개의 직접 사상 캐쉬이다. 캐쉬에 전달되는 주소는 네 개의 캐쉬 중 하나에서 그 데이터를 찾으며, 각 메모리 주소는 네 장소 중 한 곳에 저장된다. 직접 사상 캐쉬에서 하나의 위치를 두고 경쟁했던 네 메모리 항목은 이제 네 곳 중 한 곳에 자리할 수 있으며, 이 때 캐쉬는 네 항목 모두에 대해 적중한다.

2. 구조 및 동작

사원 집합연관 캐쉬는 크기가 8KB이고 그림 3에서 보는 바와 같이 128개의 라인과 해당라인의 주소 태그와 물리 주소의 태그를 비교하기 위한 비교기, 태그 인덱스를 디코딩하는 디코더, 네 개의 워드를 선택하는 멀티플렉서, Pseudo-LRU 구현기로 구성되어 있다.

한 라인은 캐쉬의 특징인 공간적 지역성에 의한 네

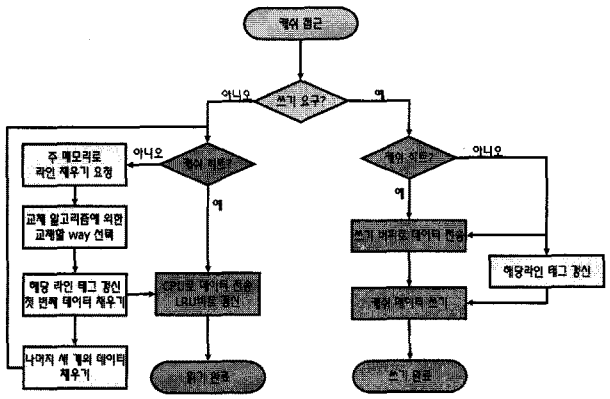

그림 4. 캐쉬 동작 흐름도
Fig. 4. Flow chart of cache operation.

개의 32비트 데이터, 참조된 주소 태그를 저장하는 21비트 태그, 라인의 유효성을 저장하는 유효비트, 최근에 참조된 집합을 저장하고 있는 3비트 LRU로 구성되어 있다.CPU로부터 메모리 참조가 발생하면 사원 집합연관 캐쉬로의 접근이 발생한다. 캐쉬는 CPU로부터 메모리의 접근목적이 읽기요구인지 쓰기 요구인지 판단한다. 만약 읽기 요구이면 사원 집합연관 캐쉬는 CPU로부터 전달받은 32비트 물리 주소의 태그 인덱스 비트를 캐쉬 내의 디코더로 디코딩하여 라인을 선택한다. 라인에 해당하는 네 개의 주소 태그 램에 저장되어 있는 각각의 21비트의 태그들과 32비트 물리주소의 물리적 영역 주소를 비교한다. 적중이 발생하면 워드 선택 비트에 의해 적중된 집합의 해당라인에 저장되어 있는 4개의 워드 중 한 개를 선택하여 CPU로 전달함과 동시에 참조된 집합을 해당라인의 LRU비트를 갱신한다. 만약 네 개의 주소 태그들과 물리적 영역 주소가 모두 일치하지 않으면 캐쉬 미스가 발생한다. 캐쉬 미스가 발생하면 라인 교체 알고리즘에 의해 결정된 교체 집합의

표 1. 4-beat wrap burst 모드
Table 1. The mode of 4-beat wrap burst.

시작 주소의 하위 2비트	4-beat wrap burst
00	0-1-2-3
01	1-2-3-0
10	2-3-0-1
11	3-0-1-2

해당 라인 데이터들을 갱신한다. 그림 4는 CPU로부터 메모리 참조가 발생하였을 때 캐쉬의 동작을 흐름도로 나타낸 것이다.

해당라인의 데이터들을 갱신할 때 캐쉬 미스가 발생한 주소의 데이터가 갱신됨과 동시에 CPU로 데이터를 전달하여 갱신 시간으로 인한 지연시간을 최소화한다. 또한 그 주위의 3개의 데이터를 WISHBONE 버스의 버스트모드 중 하나인 4-beat wrap burst 모드로 갱신한다. 4-beat wrap burst모드는 네 개의 데이터를 하나의 그룹으로 묶어 그룹 중 어떤 하나의 데이터를 갱신하면 modulo 4를 수행한 주소의 하위 2비트를 하나씩 증가시키면서 그룹의 나머지 데이터들을 갱신한다.

표 1은 시작 주소의 하위 2비트에 따른 4-beat wrap burst 모드의 적용방법을 나타낸다. 표에서 보는 바와 같이 캐쉬 미스가 발생한 주소의 하위 2비트가 01이면 하위 2비트를 1에서부터 3까지 하나씩 증가시킨 다음 하나를 증가시키면 modulo 4를 수행했기 때문에 0이 된다. 캐쉬 미스가 발생한 주소가 16진수 101이라면 먼저 101의 데이터를 갱신하고, 나머지 102, 103, 100의 데이터를 차례대로 갱신한다[10].

CPU가 메모리에 데이터를 쓰기 위해 캐쉬에 접근한 경우 캐쉬는 물리주소 상위 21비트와 해당라인의 주소 태그를 비교한다. 비교한 결과가 일치하면 쓰기 버퍼에서 전달받는 쓰기 완료 신호를 기다리고, 일치하지 않으면 해당라인의 주소태그를 물리 주소 상위 21비트로 갱신한 다음 쓰기 완료 신호를 기다린다. 물리주소와 주소태그를 비교함과 동시에 캐쉬는 물리주소와 데이터를 쓰기버퍼로 전달한다. 쓰기버퍼는 캐쉬와 주 메모리 사이에 배치하여 데이터를 메모리에 접근하여 저장하는 시간을 줄이기 위해 구현하였다. 데이터가 쓰기 버퍼에 저장되면 쓰기 버퍼는 쓰기를 완료하였다는 신호를 캐쉬로 보내고 캐쉬는 그 신호를 인식하여 해당라인의 데이터를 새로운 데이터로 갱신한다. 쓰기 버퍼는 버스가 사용되지 않을 때 주 메모리로 데이터를 전달한다.

3. 라인 교체 알고리즘

캐쉬의 라인 교체 알고리즘은 LRU, 랜덤, FIFO 등이 있지만 실제 구현은 이들 중 가장 논리적인 근거를 가지고 안정된 성능을 내는 LRU 알고리즘을 사용하고 있다. 본 논문에서 제시한 사원 집합연관 캐쉬는 라인 교체 알고리즘으로 Pseudo-LRU 알고리즘을 사용하였다. Pseudo-LRU 알고리즘은 보다 적은 비용과 면적으로 True-LRU 알고리즘의 성능을 구현하도록 고안된 알고리즘으로 다중 집합 구조의 캐쉬에서 사용되는 것으로, 각 집합별로 LRU비트를 유지하되, 몇 개의 way를 하나의 군으로 묶어 관리하는 방법이다[11~12].

사원 집합연관 캐쉬는 4개의 집합으로 구성되어 있으므로 Pseudo-LRU 알고리즘을 적용하기 위해서 Way0와 Way1을 하나의 군, Way2와 Way3을 하나의 군으로 나누어 관리하고 각 라인의 LRU는 3비트로 구현하였다. LRU의 두 번째 비트(LRU[1])는 Way0, Way1의 군과 Way2, Way3의 군을 구별하고, 첫 번째 비트(LRU[0])는 Way0와 Way1을 구별하고, 세 번째 비트(LRU[2])는 Way2와 Way3을 구별한다.

사원 집합연관 캐쉬에서의 LRU비트 갱신 방법은 표 2에서 보는 바와 같이 현재 참조된 way가 Way0나 Way1이면 LRU[1]이 1로 세팅되고 Way2와 Way3이면 LRU[1]이 0으로 세팅된다. LRU[0]은 Way0가 참조되면

표 2. LRU비트 갱신 방법
Table 2. The method of LRU bit update.

참조된 way	LRU[2]	LRU[1]	LRU[0]
Way0	-	1	1
Way1	-	1	0
Way2	1	0	-
Way3	0	0	-

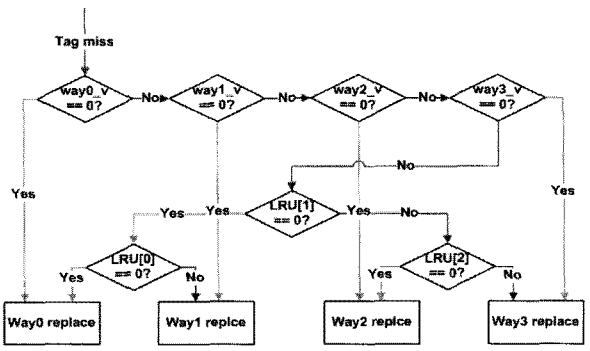

그림 5. Pseudo-LRU 알고리즘에 의한 집합 교체 방식
Fig. 5. The replacement of set by Pseudo-LRU algorithm.

1로 세팅되고 Way1이 참조되면 0으로 세팅된다. LRU[2]는 Way2가 참조되면 1로 세팅되고 Way3이 참조되면 0으로 세팅된다. 표 2에서 '-'는 LRU비트가 갱신되기 전의 값을 유지한다는 의미이다.

사원 집합연관 캐쉬는 그림 5와 같이 캐쉬 미스 발생시 Pseudo-LRU 알고리즘에 의해 교체 way를 결정한다.

그림에서와 같이 캐쉬 미스가 발생하면 각 집합의 라인의 유효성을 검사한다. 만약 집합들 중 하나라도 라인이 유효하지 않는다면 그 유효하지 않는 집합의 라인을 교체한다. 각 집합의 라인이 유효하다면 Pseudo-LRU 알고리즘에 의해 먼저 LRU비트에서 두 번째 비트(LRU[1])의 값이 '0'인지 검사하여 '0'이면 Way0과 Way1 중 하나를 교체하기 위해 LRU비트에서 첫 번째 비트(LRU[0])의 값이 '0'인지 검사한다. 첫 번째 비트의 값이 '0'이면 Way0의 해당라인을 교체하고, '1'이면 Way1의 해당라인을 교체한다. LRU비트에서 두 번째 비트의 값이 '1'이면 Way2와 Way3 중 하나를 교체하기 위해 LRU비트에서 세 번째 비트(LRU[2])의 값이 '0'인지 검사한다. 세 번째 비트의 값이 '0'이면 Way2의 해당라인을 교체하고, '1'이면 Way3의 해당라인을 교체한다.

V. 검증 및 성능 측정

본 장에서는 사원 집합연관 캐쉬의 동작을 FPGA를 이용한 에뮬레이션을 수행하여 검증한다. 또한 테스트 프로그램을 이용하여 기존의 캐쉬와 사원 집합연관 캐쉬의 미스율과 성능을 비교한다.

1. FPGA 에뮬레이션

사원 집합연관 캐쉬를 포함한 OpenRISC 코어를 FPGA로 구현하기 위해 Xilinx사의 spartan-III XC3S1000FG456 FPGA 디바이스를 사용하였고, waveplay 프로그램을 실행하여 캐쉬의 정상동작을 검증하였다. FPGA 디바이스에 구현한 하드웨어 모델은 AC97 컨트롤러 IP를 포함한 OpenRISC 기반 SoC 플랫폼이다. SoC 플랫폼은 OpenRISC 프로세서, WISHBONE 온칩버스와 주변장치들인 UART, SRAM, DMA, AC97 컨트롤러, 디버그 인터페이스, Dynalith사에서 제공하는 USB의 입출력 데이터를 JTAG의 입출력데이터로 변환해주는 OcJtagTransactor로 구성된다.

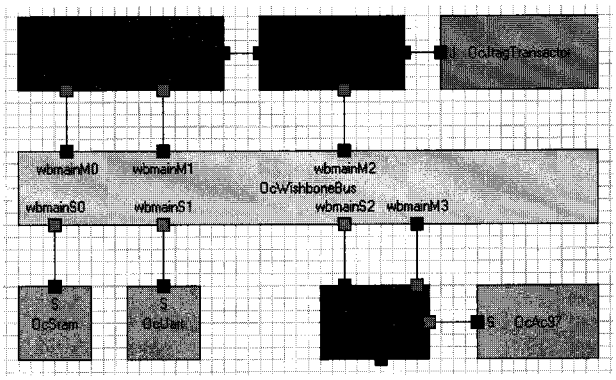

그림 6. AC97 컨트롤러를 포함한 OpenRISC 프로세서 기반 SoC 플랫폼의 구조

Fig. 6. OpenRISC based SoC platform including AC96 controller.

그림 6은 Dynalith사의 iNSPIRE-Lite 소프트웨어를 사용하여 생성한 SoC 플랫폼 하드웨어 모델이다.

사원 집합연관 캐쉬의 정상적인 동작의 검증하기 위해 Dynalith사에서 제공하는 OpenIDEA 소프트웨어를 이용하여 waveplay 프로그램을 크로스 컴파일하여 실행이미지파일을 생성하였다. Waveplay 프로그램은 SRAM에 저장되어 있는 wave 데이터를 DMA를 통해 AC97 컨트롤러로 전달하도록 디바이스들을 초기화하는 동시에 OpenRISC 프로세서 내의 Tick Timer를 이용하여 실행시간을 측정한 후 터미널 창에 실행시간을 출력한다. 검증 수행을 위해 OpenIDEA 소프트웨어의 디버거를 이용하여 타깃보드의 FPGA에 SoC 플랫폼을 다운로드하고 SRAM에 실행이미지파일을 저장한 후, waveplay 프로그램을 실행시켰다. SoC 플랫폼의 입력 클럭 주파수는 32MHz이고 UART의 보레이트는 115200Kbps이다. 그림 7은 사원 집합연관 캐쉬를 사용하여 waveplay를 실행한 결과를 나타낸 하이퍼터미널

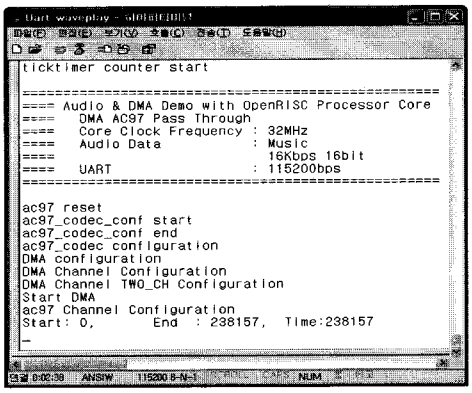

그림 7. Waveplay 프로그램 실행 결과

Fig. 7. The executed result of Waveplay program.

표 3. FPGA 구현
Table 3. The implement of FPGA.

Element	Utilization
Number of BUFGMUXs	6 out of 8 (75%)
Number of DCMs	1 out of 4 (25%)
Number of MULT18X18s	4 out of 24 (16%)
Number of RAMB16s	19 out of 24 (79%)
Clock frequency	30.3MHz

창이다.

32MHz의 클럭 주파수로 동작하는 OpenRISC 프로세서를 포함한 SoC 플랫폼에서 waveplay 프로그램을 실행하였을 때 Tick Timer를 이용하여 실행 클럭수를 카운트하고 AC97 컨트롤러 및 DMA를 초기화한 후 실행시키는 것을 하이퍼터미널 창으로 확인하였고 사원 집합연관 캐쉬를 사용한 waveplay 프로그램의 실행 클럭수는 238157이다. 사원 집합연관 캐쉬를 FPGA에 구현하기 위해 사원 집합연관 캐쉬를 포함한 OpenRISC 코어를 생성하였다. 하드웨어 모델을 생성한 다음 Xilinx사에서 제공하는 XST를 이용하여 synthesis를 수행한 다음, Xilinx사의 ISE 툴을 이용하여 P&R을 수행하여 생성된 2진 파일을 FPGA에 다운로드한다. 표 3은 AC97 컨트롤러를 포함한 OpenRISC 프로세서 기반 SoC 플랫폼을 FPGA로 구현한 결과이다.

2. 성능 비교

기존의 직접사상 캐쉬와 사원 집합연관 캐쉬의 성능을 비교하기 위해 H.264 복호기의 역변환 및 역양자화 기능 블록을 테스트프로그램으로 사용하였다. H.264 복호기의 역변환 및 역양자화 기능 블록은 16x16 크기의 매크로 블록을 처리하는 동안 역양자화 및 역변환을 순차적으로 26번 수행한다. 그림 8은 역양자화를 수행한 후 역변환을 수행하기 전까지 실행시간을 기준으로 기존의 캐쉬를 포함한 코어와 사원 집합연관 캐쉬를 포함한 코어의 실행 사이클 수를 나타낸다.

첫 번째 역양자화를 수행하고 역변환을 수행하는 동안 기존의 코어와 사원 집합연관 캐쉬를 포함한 코어의 실행 사이클 수는 234개로 동일하였으나, 역양자화 및 역변환을 반복적으로 수행한 경우 사원 집합연관 캐쉬를 포함한 코어의 실행 사이클 수는 111개로 감소하였고, 기존 코어의 실행 사이클 수는 206개로 감소하였다. 또한 18번 째 역양자화를 수행한 후 Hadamard 역변환을 수행한 후 기존 코어의 실행 사이클 수는 91개로 일

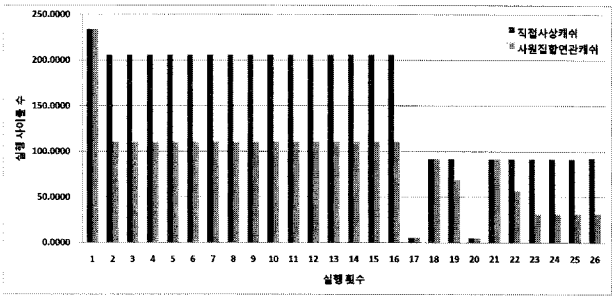

그림 8. 역양자화 후 역변환에 대한 실행 사이클 수 비교

Fig. 8. The comparison of the number of execution cycle for the inverse transform after inverse quantization.

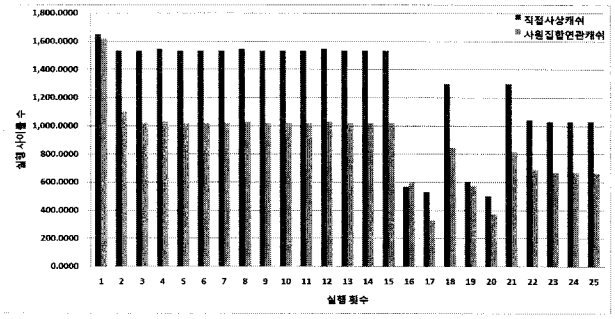

그림 9. 역변환 후 역양자화에 대한 실행 사이클 수 비교

Fig. 9. The comparison of the number of execution cycle for the inverse quantization after inverse transform.

정하였고, 사원 집합연관 캐쉬를 포함한 코어의 실행 사이클 수는 91개에서 31개로 감소하였다. 따라서 사원 집합연관 캐쉬를 포함한 코어의 성능이 기존의 코어에 비해 86%이상 향상했다.

그림 9는 역변환을 수행한 후 역양자화를 수행하기 전까지 코어가 실행한 사이클 수를 나타낸다. 역변환 후 역양자화를 반복해서 수행하는 동안 사원 집합연관 캐쉬를 포함한 코어의 실행 사이클 수는 1,019개이고, 기존 코어는 1,533개이다. 또한 Hadamard 역변환을 수행한 후 기존 코어의 실행 사이클 수는 1,029개이고 사원 집합연관 캐쉬를 포함한 코어는 668개이다. 사원 집합연관 캐쉬를 포함한 코어의 성능이 기존의 코어보다 50%이상 향상되었다.

그림 10은 역양자화를 수행한 후 역변환을 수행하기 전까지의 실행시간을 기준으로 기존 코어와 사원 집합연관 캐쉬를 포함한 코어의 미스율을 나타낸다. 첫 번째 실행에서 기존 코어와 제안한 코어의 미스율은 20.37%로 동일하였으나, 반복 횟수가 증가할수록 제안

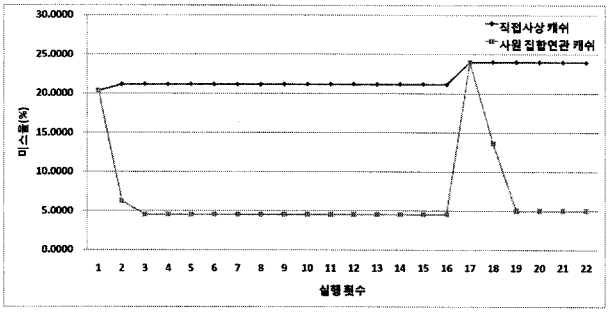

그림 10. 테스트 프로그램에 대한 미스율 비교
Fig. 10. The comparison of the miss ratio for test program.

한 코어의 미스율은 4.5%로 감소하였고, 기존 코어는 21.15%로 증가하였다. 또한 Hadamard 역변환을 수행하였을 때 기존 코어와 제안한 코어의 미스율은 24%로 동일하였으나, 횟수가 증가하면서 제안한 코어의 미스율은 5%로 감소하였고, 기존 코어의 미스율은 24%로 증가하였다. 따라서 제안한 코어의 미스율은 기존 코어의 미스율보다 15%이상 감소했다.

VI. 결 론

본 논문에서는 OpenRISC 코어의 성능 향상을 위해 사원 집합연관 캐쉬를 제안하였다. 사원 집합연관 캐쉬는 직접 사상 캐쉬의 단점인 높은 접근 실패율을 줄이기 위해 한 개의 블록에 네 개의 메모리 블록이 사상되는 구조로 되어 있다. 제시한 캐쉬의 교체 알고리즘으로는 Pseudo-LRU 알고리즘을 사용하여 적은 면적으로 True-LRU 알고리즘의 성능을 구현하였다. 또한 캐쉬 미스 발생시 발생하는 라인의 데이터 갱신 시간으로 인한 지연시간을 줄이기 위해 미스가 발생한 주소의 데이터를 갱신함과 동시에 CPU로 데이터를 전달함으로써 지연시간을 최소화하였다. FPGA 에뮬레이션을 이용하여 사원 집합연관 캐쉬를 포함한 OpenRISC코어의 정상 동작을 검증하였고, 테스트 프로그램을 이용하여 성능을 측정한 결과, 기존의 코어에 비해 사원 집합연관 캐쉬를 포함한 코어의 성능은 50% 이상 향상했고, 미스율은 15% 이상 감소했다. 따라서 기존의 직접사상 캐쉬 대신 사원 집합연관 캐쉬를 사용하는 것이 코어의 성능 향상 및 메모리의 접근시간이 개선된다는 결론을 이끌어냈다.

참 고 문 헌

[1] 김동욱, 이준원, 박승규, "고속 RISC 프로세서를 위한 개상 캐쉬 구조", 정보과학회논문지, 제 23권 A편, 제9호, 887-898쪽, 1996년 9월
[2] David A. Patterson, John L. Hennessy, "Computer Organization and Design, 3rd Edition", Morgan Kaufmann Pub, p. 594, 2004.
[3] William Stallings, "Computer Organization and Architecture, 7th Edition", PrenticeHall, p.754, 2005.
[4] 경종민, 박인철, "고성능 마이크로프로세서 구조 및 설계 방법", 대영사, p. 475, 2000.
[5] Mark D. Hill, "A case for direct-mapped caches", Computer, IEEE JNL, Volume 21, Issue 12, pp. 25-40, December, 1998.
[6] Chenxi Zhang, Xiaodong Zhang, Yong Yan, "Two fast and high-associativity cache schemes", Micro, IEEE JNL, Volume 17, Iss 5, pp. 40 - 49, Sept.-Oct. 1997.
[7] Hsin-Chuan Chen, Jen-Shiun Chiang, "Design of an adjustable-way set-associative cache", IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, Volume 1, pp. 315-318, August, 2001.
[8] Damjan Lampret, OpenRISC1200 IP Core Specification Revision 0.7, p. 63, 2001.
[9] Damjan Lampret, OpenRISC1000 Architecture Manual, p. 343, 2003.
[10] Richard Herveille, WISHBONE System-On-Chip (SoC) Interconnection Architecture for Portable IP Cores, p. 140, September, 2002.
[11] 이종익, 손승일, 이문기, "캐쉬 메모리에서 True-LRU 알고리즘과 Pseudo-LRU 알고리즘의 성능 비교", 정보과학회논문지 제 23권 제11호, 1996. 11
[12] Nikitas Alexandridis, "Design of Microprocessor-based systems", Prentice Hall, p. 541, 1993.

저 자 소 개

정 홍 균(학생회원)
2007년 한밭대학교 정보통신공학과 학사
2007년～현재 한밭대학교 정보통신공학과 석사과정
<주관심분야 : 임베디드 프로세서, SoC 플랫폼 설계, 하드웨어/소프트웨어 통합설계, 멀티미디어 코덱 설계>

류 광 기(평생회원)
1986년 한양대학교 전자공학과 학사
1988년 한양대학교 전자공학과 석사
2000년 한양대학교 전자공학과 박사
1991년～1994년 육군사관학교 전자공학과 전임강사
2000년～2002년 한국전자통신연구원 시스템 IC 설계팀
2003년～현재 한밭대학교 정보통신공학과 조교수
<주관심분야 : SoC 플랫폼 설계 및 검증, 하드웨어/소프트웨어 통합설계 및 통합검증, 멀티미디어 코덱 설계>