고성능 DSP에서 동영상 인코더의 최적화 구현을 위한 캐쉬 및 내부 메모리 성능 분석

Performance Analysis of Cache and Internal Memory of a High Performance DSP for an Optimal Implementation of Motion Picture Encoder

임세훈, 정선태
숭실대학교 정보통신전자공학부

SeHun Lim(nowboy@su.ac.kr), Sun-Tae Chung(cst@ssu.ac.kr)

요약

고성능 DSP는 보통 캐쉬와 내부 메모리를 지원한다. 이러한 고성능 DSP에 멀티미디어 스트림 응용을 최적화하여 구현하고자 하는 경우에는, DSP 가 지원하는 캐쉬와 내부 메모리를 효율적으로 잘 활용하여야 한다.

본 논문에서는 2단계 레벨 캐쉬 구조 및 내부 메모리 구성을 지원하는 고성능 DSP인 TMS320C6000 시리즈에 대해 동영상 인코더와 같은 멀티미디어 스트림 처리 응용을 최적으로 구현하기 위해서 필요한 캐쉬 성능 분석, 내부 메모리 구성 및 배치에 따른 성능 분석과 개선 방안에 대해 연구하였다.

분석 및 실험 결과, L2 메모리의 경우, 이중 집합연관 캐쉬로 구성하고, 남은 메모리는 내부 메모리로 구성하는 것이 수행 시간 성능 개선에 효과적임을 확인하였다. 또한, L1P 캐쉬의 경우는 자주 호출되고 시간이 많이 소요되는 루틴들을 연속적으로 내부 메모리에 배치하는 것이 L1P 캐쉬의 히트 율을 개선하며, L1D 캐쉬의 경우는 사용하는 데이터의 크기를 조절하므로 써 쉽게 히트 율을 개선할 수 있다는 것을 밝혔다.

본 논문의 연구 결과는 고성능 DSP 에 멀티미디어 스트림 처리 응용을 최적화로 구현하는데 도움을 줄 것으로 기대한다.

■ 중심어 : | 고성능 DSP | 동영상 코덱 | 캐쉬 | 내부 메모리 | SW 최적화 |

Abstract

High Performance DSP usually supports cache and internal memory. For an optimal implementation of a multimedia stream application on such a high performance DSP, one needs to utilize the cache and internal memory efficiently.

In this paper, we investigate performance analysis of cache, and internal memory configuration and placement necessary to achieve an optimal implementation of multimedia stream applications like motion picture encoder on high performance DSP, TMS320C6000 series, and propose strategies to improve performance for cache and internal memory placement.

From the results of analysis and experiments, it is verified that 2-way L2 cache configuration with the remaining memory configured as internal memory shows relatively good performance. Also, it is shown that L1P cache hit rate is enhanced when frequently called routines and routines having caller-callee relationships with them are continuously placed in the internal memory and that L1D cache hit rate is enhanced by the simple change of the data size.

The results in the paper are expected to contribute to the optimal implementation of multimedia stream applications on high performance DSPs.

■ keyword : | High Performance DSP | Motion Picture Encoder | Cache | Internal Memory | SW Optimization |

* 본 연구는 숭실대학교 교내연구비 지원으로 이루어졌습니다.
접수번호 : #080418-002
접수일자 : 2008년 04월 18일
심사완료일 : 2008년 05월 09일
교신저자 : 정선태, e-mail : cst@ssu.ac.kr

I. 서 론

외부 메모리와 프로세서와의 속도 차이를 극복하기 위해 프로세서 내부에 캐쉬(cache)가 개발되어 왔다[1]. 한편, 캐쉬가 갖는 파워 소모량 때문에 내부 SRAM이 대안으로 연구 개발되었다[2][3]. 캐쉬와 내부 메모리가 동시에 프로세서 내부에 존재하는 것이 공정상으로도 가능하므로 많은 임베디드 프로세서 (모토롤라 68HC12, TMS320C6x, 등)가 내부 캐쉬 및 내부 메모리를 프로세서 내부에 모두 지원하고 있다.

디지털 신호 처리 연산을 효과적으로 하는 고성능 DSP 코어와 멀티미디어 응용 개발에 유용한 각종 I/O을 칩 내부에 지원하고 있어 임베디드 멀티미디어 스트림 처리 응용의 ASSP (Application Specific Standard Product)로 많이 사용되고 있는 고성능 DSP 인 TMS320C6000 시리즈 칩 계열에서도, L1, L2 의 2단계 캐쉬뿐만 아니라, 내부 메모리를 지원하고 있다[4].

이러한 고성능 DSP에 멀티미디어 스트림 응용을 개발하는 경우, 해당 DSP가 제공하는 캐쉬 구조를 효율적으로 이용하고[5], 외부 메모리보다 접근 속도가 빠른 내부 메모리에 효율적으로 코드 및 데이터를 배치하여야 좋은 성능을 얻을 수 있다.

본 논문에서는 2단계 레벨 캐쉬 구조, L1P, L1D, L2 캐쉬/내부메모리 구성을 지원하는 고성능 DSP인 TMS320C621x/671x/64x 패밀리에 대해 멀티미디어 스트림 처리 응용을 최적으로 구현하기 위해서 필요한 캐쉬 성능 분석, 내부 메모리 구성 및 배치에 대한 성능 분석 및 성능 개선 방안에 대한 연구를 수행하였다.

보다 구체적인 성능 분석을 위하여 고성능 DSP로는 TMS320C6713[6]를, 멀티미디어 스트림 응용으로는 UBC의 H.263 인코더[7]를 이용하였다. H.263 은 MPEG 1/2/4, H.264 등의 다른 동영상 코덱과 인터모드 코딩에서 움직임 추정 및 보상을 이용한다는 점에서 기본적인 원리는 동일하며 다만 세세한 특징이 좀 차이가 있을 뿐이다[8].

분석 및 실험 결과, L2 캐쉬는 2중 집합연관 캐쉬로 구성하고, 남는 용량은 내부 메모리로 이용하는 것이 효율적임을 확인하였다. 또한, TMS320C671x의 L1P 캐쉬는 직접사상 캐쉬이므로, 자주 사용되고 계산 량이 많은 루틴을 중심으로 호출관계가 있는 루틴을 메모리에 연속으로 배치함으로써, 충돌 캐시 미스를 줄이고 캐쉬 히트 율이 높일 수 있음을 확인하였다. L1D 캐쉬의 경우, 간단한 데이터 타입 조정으로도 캐쉬 히트 율을 개선할 수 있음을 보였다.

캐쉬 성능 최적화에 대한 연구로는 [9-13] 등이 있다. 그러나 이러한 연구들은 일반적인 캐쉬 최적화에 대한 분석이다. TMS320C6000 에서 H.263 인코더 구현에 대한 연구로는 [14][15] 가 있다. 그러나 캐쉬 분석은 주어지지 않았다.

이들 연구와는 달리, 본 논문의 연구는 실제적인 고성능 DSP인 TMS320C671x 상에서 구체적인 멀티미디어 스트림 처리 응용인 H.263 인코더의 최적화 구현을 위해 캐쉬 및 내부 메모리 구성 및 배치에 대해 실질적인 분석 및 개선 방안이 연구되었다는 점에서 의의가 있다.

내부 메모리 최적화에 대한 연구, 즉 코드 및 데이터 등을 내부 메모리에 어떻게 배치하는 게 바람직한 가에 대한 연구는 많이 진행되어 왔다[2][3]. 이러한 연구들은 모두 임베디드 프로세서 안에 메모리가 캐쉬와 내부 메모리로 구성되는 것만을 고려한 것으로 본 논문의 TMS320C6713처럼 캐쉬 계층 구조(L1P, L1D, L2)와 내부 메모리를 갖는 경우와는 다르며, L2 메모리를 L2 캐쉬와 내부 메모리로 분할 구성하는 문제는 고려되지 않았다.

본 논문의 구성은 다음과 같다. 제2절에서는 본 논문 내용의 배경인 캐쉬, 내부 메모리, TMS320 C671x 패밀리의 캐쉬/내부메모리 구조, 멀티미디어 응용 특성에 대해 간단히 살펴본다. 제3절에서는 TMS320C6713에서의 H.263 인코더 구현을 위한 캐쉬 분석 및 개선 방안 결과가 기술된다. 제4절에서는 실험 및 검토 결과가 설명되며, 마지막으로 제5절에 결론이 주어진다.

논문의 배경인 H.263에 대해서는 [8]을 참조하라.

II. 기술적 배경

1. 캐쉬 개요

캐쉬는 속도 성능개선이 매우 빠른 CPU와 이를 따라가지 못하는 메인 메모리 사이의 속도 차이를 극복하기 위해 고안된 고속 메모리이다. 캐쉬는 계층 구조를 가질 수 있다. 즉, CPU에서 가장 가까운 캐쉬는 L1 캐쉬, 그 다음에 위치한 캐쉬는 L2 캐쉬 등으로 명명된다. CPU는 읽고자 하는 명령어나, 읽거나 갱신하고자 하는 데이터를 먼저 L1 캐쉬에서 참고하고 존재하는 경우(캐쉬 히트), L1 캐쉬에서 가져오거나 L1 캐쉬에 갱신한다. 만일 L1 캐쉬에 없으면(캐쉬 미스), 다음 캐쉬(L2)를 참조하거나, 다음 단계 메모리(내부 메모리 또는 외부 메인 메모리)를 참조하게 된다. 캐쉬 미스가 발생하면 다음 단계 캐쉬 또는 메모리를 참조하여야 하므로 명령 수행 완료에 필요로 하는 CPU 클록 사이클이 증가하게 되어 해당 프로그램 수행 시간을 증가시키는 주요 요인이 된다. 하위 단계 캐쉬(또는 메모리)에서 한 번에 가져오는 정보(명령어 또는 데이터) 단위를 라인(또는 블록)이라 한다.

가져온 하위 단계 메모리 라인이 위치할 장소가 한 곳으로만 고정된 경우의 캐쉬를 직접사상 캐쉬(direct mapped cache), 둘 이상(2,3,4 등)의 장소를 갖는 캐쉬를 2중/3중/4중 집합연관 캐쉬(2/3/4-way set-associative cache), 임의의 장소로 배치될 수 있는 캐쉬를 완전연관캐쉬(fully associative cache) 라 한다. 캐쉬 미스의 발생 원인은 캐쉬 메모리를 처음 참조하기 때문에 발생하거나(첫참조미스, compul sory miss), 캐쉬의 저장 공간이 부족하거나(용량미스, capacity miss), 캐쉬 내 다른 공간은 충분함에도 불구하고 배치될 장소에 다른 메모리 내용이 들어가 있어 충돌이 발생하게 되는(충돌미스, conflict miss) 등의 3가지에 원인에 기인한다.

캐쉬에 대한 보다 자세한 내용은 [1]을 참조하기로 한다.

2. TMS320C621x/671x/64x 패밀리 캐쉬 구조 및 동작

TMS 320C6000 칩 패밀리[4]는 Texas Instruments 사의 VLIW(Very Long Instruction Word) 기반 고성능 DSP 프로세서들로 621x/64x 계열은 고정 소수점 연산을, 671x 계열은 부동소수점 연산을 지원한다.

TMS320C621x/671x/64x 패밀리의 캐쉬는 L1, L2 의 2계층 구조를 가지며, L1 및 L2 캐쉬 모두 프로세서 내부에 위치한다. L1 캐쉬는 직접사상 방식의 프로그램용 L1P 캐쉬와 2중 집합연관 방식의 데이터용 L1D 캐쉬의 2가지로 구성된다. C621x/C671x(C64x)에서 L1P 와 L1D 캐쉬의 크기는 각각 4KB씩(C64x의 경우는 16KB씩)이다. 라인 크기는 L1P 캐쉬의 경우 64바이트(32바이트), L1D 캐쉬는 32바이트(64바이트)이다. L2 캐쉬는 프로그램 및 데이터용으로 모두 사용되는 통합캐쉬로 용량은 최대 64KB(256KB)이며, 라인 크기는 128 바이트이다. L2 캐쉬는 구성에 따라, C621x/C671x 에서는 L2 캐쉬 없음(0KB), 직접사상 (16KB), 2중집합연관(32KB), 3중집합연관(48KB), 4중 집합연관(64KB) 등으로 구성될 수 있으며, C64x 에서는 32/64/128/256KB의 4가지 구성이 가능하고 모두 4중 집합연관 구조이다. 최대 64KB(256KB) 중 L2 캐쉬로 구성되지 아니한 L2 메모리는 내부 메모리로 구성된다. TMS320C6713의 경우는 별도의 192KB, C64x 는 768KB의 별도의 내부 메모리를 가지고 있다. 외부 메모리는 EMIF (External memory Interface)를 통해 CPU 및 CPU 내부 메모리와 연결된다.

L1P, L1D 캐쉬 접근 속도는 1 CPU 클록 사이클이며, L2 캐쉬 및 내부 메모리 접근에 4 CPU 클록 사이클이 필요하다. 논문의 실험에서 사용한 C6713은 225MHz 이므로, 1 CPU 클록 사이클은 4.4ns 이다. 외부 메모리의 경우, 보통 100MHz SDRAM 을 지원하는 데, 읽는 데 최소 6 메모리 사이클(60ns), 쓰는 데, 최소 4 메모리 사이클(40ns)이 소요된다. 따라서 L2 메모리에서 명령어나 데이터를 읽어 오는 것이, 외부 메모리에서 읽어 오는 것보다 훨씬 빠름을 알 수 있다.

3. 멀티미디어 응용 특성

멀티미디어 응용의 경우에 처리할 데이터는 연속적이고, 이를 처리하는 명령어는 반복적(loop)이다[11].

즉, 멀티미디어 응용에서 데이터는 주로 한번 사용되면 다시는 사용되지 않는 재사용성이 적은 스트림 데이터이고, 명령어는 다시 사용될 명령어가 많다는 것이 특징이다. 명령어의 분포는 메모리 참조 명령, 분기 명령, 산술/논리 연산 명령 순이다[10]. 또한 멀티미디어 응용은 규칙적이고 예측 가능한(predictable) 제어 흐름을 나타내는 것으로 잘 알려져 있다[10]. 따라서 캐쉬 미스 및 분기 예측 에러에 의한 페널티는 전체 수행 시간에 크게 영향을 끼치지 않는다.

데이터의 경우, 주소(pointer) 데이터 경우 외에 16비트 이상 데이터는 적으며, 8비트 데이터가 제일 많고 32비트 데이터(주소 포인터), 16비트 데이터 순서이다.

III. TMS320C6713에서 효율적인 H.263 인코더 최적화를 캐쉬 분석

1. TMS320C6713 L2 캐쉬/내부 메모리 분석

1.1 L2 캐쉬 구성/내부 메모리 구성 분석

L2 캐쉬 구성은 TMS320C621x/671x의 경우, 구성가능 한 64KB의 L2 메모리를 0KB, 16KB 직접사상 캐쉬, 32KB 2중, 48KB 3중, 64KB 4중 집합연관 L2 캐쉬 메모리로 구성하는 것을 말한다. 구성하고 남은 L2 메모리는 내부 메모리로 사용된다. 따라서 이 경우 L2 내부 메모리는 64KB, 48KB, 32KB, 16KB, 0KB 가 증가된다.

TMS320C621x/671x의 경우, L1P 캐쉬는 4KB 직접사상 캐쉬로 32 바이트 크기 명령어 패킷 128개를 한번에 보유할 수 있다. 프로파일링 결과 (4.2절 참조), 가장 시간이 많이 소요되고 많이 호출되는 Sad_Macroblock() 루틴의 경우 최적화하는 경우 반복되는 명령어 패킷은 총 50개 미만이다[14]. 이 경우 모든 반복 명령어 패킷이 L1P 캐쉬 안에 포함되어 수행될 수 있다. 그러나 만일 반복 되는 루틴의 명령어 패킷 수가 128개 이상인 경우 캐쉬 미스는 피할 수 없게 된다. 이 경우, L2 캐쉬가 없으면 L1P 의 라인 사이즈는 64바이트 이므로 2개 명령어 패킷이 수행 때마다 다시 L1P 캐쉬 미스가 발생한다. L2 캐쉬가 구성된 경우, L2 캐쉬의 라인 사이즈는 128바이트 이므로 일단 L2 캐쉬에 들어오면 다른 3개의 명령어 패킷 수행까지는 L2 캐쉬 미스가 발생하지 않으며, L2 캐쉬 사이즈에 따라 충분히 많은(128개보다 많은) 연속된 명령어 패킷을 보유할 수 있어, L2 캐쉬 미스는 많이 줄일 수 있다.

L1D 캐쉬의 경우 4KB 용량의 2중 집합연관 캐쉬로 각 데이터 패스 당(TMS320C6000 은 2개의 데이터통로가 있음) 2KB 용량으로 4 바이트 크기 데이터(정수 타입 데이터의 경우) 512개를 보유할 수 있다. 블록 매칭을 수행하는 SAD_Macroblock() 루틴의 경우, 1 매크로블록 당 256개의 정수(UBC H.263에서는 최종적 매크로블록에 담아질 데이터 타입으로 정수를 사용함) 데이터가 필요하고, 참조 프레임(이전 복원된 프레임)에서 비교되어야 할 탐색 영역은 최대 46×46개 정수(2116개 정수) 데이터가 필요하다. 즉, 한번의 SAD_Macroblock() 루틴 수행 시에 최대 2372개의 정수 데이터가 소요된다. 따라서 4KB(데이터패스 당 2KB) L1D 캐쉬로는 SAD_Macrblock() 함수를 수행하는 데 필요한 데이터를 모두 보유하기에는 충분하지 않아, SAD_MAcroblock() 루틴 수행 중에 L1D 캐쉬 미스가 불가피하다. L2 캐쉬가 지원되지 않으면 L1D 라인 사이즈가 32바이트 이므로 8개의 정수 데이터 작업 후 다시 캐쉬 미스가 발생하게 된다. 반면, L2 캐쉬의 경우 라인 사이즈는 128바이트 이므로 32개의 정수 데이터 작업까지 가능해진다.

이상의 L1P와 L1D 캐쉬 분석에 따라 L2 캐쉬 구성의 필요성은 분명해진다.

그러면, L2 캐쉬는 얼마로 구성하는 것이 좋을까?

L2 캐쉬는 통합 캐쉬이므로 데이터 보유를 지원하고, TMS320C6713의 경우 2개의 데이터패스가 지원되므로 동시 작업을 충분히 지원하기 위해서, L2 캐쉬도 쌍으로 구성하는 게 좋을 것으로 판단된다. 따라서 L2 캐쉬는 2중 또는 4중 집합연관 캐쉬로 구성하는 게 필요한데, 4중 집합연관 캐쉬의 경우 L2 구성 가능 메모리 64KB 모두를 L2 캐쉬로 사용하는 게 된다. 멀티미디어 응용은 반복되는 짧은 루틴, 짧은 시간에 한 번 쓰고 버리는 스트림 데이터를 사용한다는 특성과 앞의 L1P와 L1D 캐쉬 분석을 통해 L2 캐쉬 용량이 꼭 매우 클 필

요가 없다는 것을 알 수 있다. 이러한 분석은 [9]에서는 멀티미디어 프로세서의 경우, L2 캐쉬 크기는 크게 영향을 미치지 않는다고 보고하고 있는 것과 부합한다.

따라서 L2 캐쉬는 32KB 2중 집합연관 캐쉬로 구성하는 게 적절하다. 이는 4.2절의 실험으로 확인된다.

1.2 L2 내부 메모리 배치

UBC H.263 인코더에서 내부 메모리 (총 224KB)에 어떤 루틴과 어떤 데이터를 배치하는 게 좋을까? 또한 스택 세그먼트와 동적 메모리의 배치는 어떻게 하는 게 좋을까?

먼저, 동적 메모리 배치를 살펴보자. UBC H.263 인코더 구현에서 필요로 하는 동적 메모리는 500KB가 넘는다. 따라서 동적 메모리는 외부 메모리에 할당할 수밖에 없다.

다음, 스택세그먼트의 배치를 검토해보자. 스택 세그먼트의 경우 현재의 컴파일러/링커가 분산 스택 세그먼트를 지원하지 않으므로 단일 스택 세그먼트만을 고려하여야 한다. 스택 세그먼트는 스택 프레임으로 구성되는데, 각 스택 프레임은 불리어질 함수로 전달될 함수 인자(TMS320C6x DSP 경우는 10개까지의 인자는 레지스터 이용하여 전달함), 복귀주소(TMS320C6x DSP는 레지스터 이용), 저장되어야 할 레지스터 값들, 해당 함수의 지역 변수 등을 저장한다. 현재 UBC H.263 인코더 구현에서는 최대 6 호출 깊이(main() 함수로부터 연속적으로 불리어져 중첩되는 함수 개수의 최대)를 갖는 데, 최대 6개의 스택 프레임을 모두 고려하더라도 총합의 스택 세그먼트 값은 12KB 미만이다(이는 실험을 통해서 확인하였다). 따라서 내부 메모리(총 224KB)에 배치가 가능하다.

UBC H.263 인코더의 경우 데이터 세그먼트는 .bss(Unintialized global data) 섹션 1768바이트,const 섹션 12286바이트 포함하여 15KB 미만이다. 역시 내부 메모리에 배치가 가능하다.

UBC H.263 인코더의 경우 코드(텍스트) 세그먼트는 .text 섹션만을 따지면 250KB 가까이 된다. 따라서 코드 세그먼트 전체가 내부 메모리에 배치 될 수 없다. 따라서 자주 사용되고 시간이 많이 소요되는 루틴만을 선택적으로 내부 메모리에 배치되어야 한다. 또한, 캐쉬 미스를 줄이기 위해 호출 관계(caller-callee 관계)가 있는 루틴들도 가급적 연속적으로 내부 메모리에 배치하는 게 필요하다.

본 논문의 연구에서는 UBC H.263 인코더의 프로파일링 실험 (4.2절)을 통해 가장 많은 시간이 소요되며 가장 많이 호출되는 루틴들을 조사했다. 이들 루틴과 호출관계에 있는 루틴들을 내부 메모리에 배치하여 내부 메모리 활용 성능을 개선할 수 있었다.

2. TMS320C6713 L1P 캐쉬 분석

동영상 코딩과 같은 멀티미디어 스트림 처리 응용의 경우, 처리 데이터는 매번 바뀌나 같은 종류의 반복 작업 처리가 많기 때문에 한 번 캐싱된 명령어들은 조만간 재사용될 가능성이 매우 높다. 따라서 조만간 재사용될 가능성이 있는 동안에는 해당 L1P 캐쉬 라인의 방출이 적게 발생하도록 노력하여야 한다. 그런데, TMS320C6000 시리즈의 L1P 캐쉬는 직접사상 방식이기 때문에 다른 n-way (n=2,3,4, ...) 집합연관 방식의 캐쉬보다 충돌 가능성이 높다. 따라서 멀티미디어 스트림 처리 응용의 경우, L1P 캐쉬의 메모리 액세스 시간 개선(감소)에는 충돌 캐쉬 미스를 줄이는 게 매우 중요하다. 현재의 컴파일러와 링커는 자동적으로 캐쉬 충돌에 대한 고려를 하지 못하기 때문에, 링커에 의한 부적절한 프로그램 코드들의 메모리 주소 배치가 프로그램 수행 중에 충돌 캐쉬 미스를 초래할 수 있다. 예를 들어, 동일한 L1P 캐쉬 라인에 매핑 되는 L2 메모리 또는 외부 메모리 내에 배치되어 있는 2개의 함수 function_1, function_2가 {function_1, function_2} 순으로 계속 반복 수행되어야 하는 경우, 만일 2개의 함수가 L1P 캐쉬에서 겹쳐지는 캐쉬 라인으로 매핑되어 캐싱된다고 하면, 2개의 함수를의 명령어들을 보유할 L1P 캐쉬의 용량이 충분함에도 불구하고, 반복적으로 2개의 함수가 연달아 호출되는 경우에 충돌 미스를 계속 반복 되는 것을 피할 수없게 된다[그림 1].

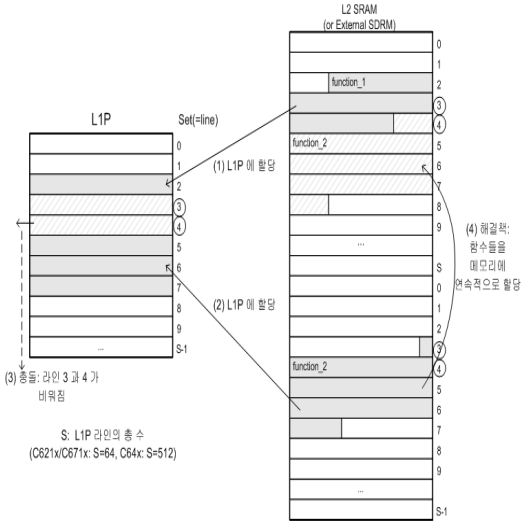

그림 1. 충돌 캐쉬 미스의 예

이러한 충돌 캐쉬 미스 경우는, 링커 옵션을 조정하여 충돌되는 함수들을 내부/외부 메모리에 연속적으로 배치하도록 하면, L1P 캐쉬에 캐쉉될 때, 다른 라인으로 캐쉉되기 때문에 충돌을 방지할 수가 있고 따라서 충돌 캐쉬 미스 율을 크게 개선시킬 수 있다.

즉, 호출 관계를 고려하여 호출관계에 있는 함수는 메모리에 연속적으로 배치하도록 하여야 한다.

UBC H.263 인코더의 프로파일링 결과 (4.2절), SAD_Macrobrock(), MotionEstimation(), Idct(), Quant_blk(), DCT(), CodeCoeff(), LoadArea(), DeQuant() InterpolateImage(), FindHalPel(), ReconImage(), MB_Decode(), Quantization() 등의 루틴들이 자주 수행되면서도 자체 루틴 수행시간이 많이 소요되었다. 이 루틴들 가운데, SAD 루틴(SAD_Macrobrock())이 가장 많은 시간이 소요되며 또한 가장 많이 호출된다. 이 SAD 루틴은 움직임 추정 루틴에서 호출된다. 따라서 SAD 루틴과 움직임 추정 루틴(MotionEstimation())을 반드시 내부 메모리에 연속적으로 배치시킬 필요가 있다.

본 논문 연구의 실험에서는 이와 같이 호출 관계가 있는 루틴들 중 가장 자주 불리워지고 시간이 많이 소요되는 루틴들의 순서로 호출 관계의 루틴들이 연속적으로 내부 메모리에 배치되도록 링커를 조절하였다. 이러한 방법을 통해 캐쉬 미스 율이 개선됨을 실험을 통해 확인할 수 있었다.

3. TMS320C6713 L1D 캐쉬 분석

기본적으로 데이터 캐쉬 미스를 줄이는 데에는 데이터의 주소 배치를 가급적이면 캐쉬에서 충돌이 잘 나지 않도록 하며(data placement), 일단 캐쉬로 들어 온 데이터를 가능한 한 재사용(reuse)하고, 필요한 데이터를 캐쉬로 불러들일 때는 가장 나중에 사용될(least recently used) 데이터를 캐쉬에서 방출하고 이 자리를 사용하도록 하는 방안과 사용될 데이터를 캐쉬로 미리 가져오는 전략(프리페칭)을 고려하여야 한다.

그러나 앞서 기술한 멀티미디어 응용의 데이터는 재사용성이 적은 스트림 데이터이므로, 재사용 전략은 필요하지 않다. 소프트웨어 캐쉬 프리페칭의 경우, 매 프리페칭시에 첫 참조 캐쉬 미스가 발생하며 이때 캐쉬로 가져 오는 데이터 크기 역시 라인 사이즈이므로, 필요할 때 마다 페칭하는 요구페칭(demand fetching)과의 캐쉬 미스 페날티의 차이는 별로 없다. 또한, TMS320C6000 시리즈의 경우, 캐쉬 프리페칭을 하드웨어적으로 지원하지 않으므로.프리페칭 전략은 별로 소용이 없다.

TMS320C6000 시리즈에서 L1D 캐쉬는 2중 집합연관이므로 직접사상 캐쉬 방식의 L1P보다는 충돌 미스 발생이 줄어든다.

따라서 멀티미디어 스트림 처리 응용의 경우 L1D 캐쉬의 캐쉬 미스율 개선을 위해서는 용량 미스를 줄이는 것이 중요하다. 이를 위하여, 사용되는 데이터 타입을 조사하여 요구되는 메모리 대역폭 요구를 감소시키는 방안의 검토가 필요하다. 멀티미디어 스트림 처리의 사용되는 정수 데이터 크기는 대부분 8비트, 16 비트인 것으로 알려져 있다. 또한 실수의 경우, 32비트 단정도(float)나 64 비트 배정도(double)가 사용된다.

본 논문에서는 먼저 H.263 인코더의 일부분 함수에 대해, C 언어의 double(64 비트)로 정의된 실수 데이터를 모두 단정도(32비트)로 변경한 경우에 캐쉬 미스 율을 테스트해보았다. 영상 품질을 측정하는 PSNR(Peak

Signal to Noise Ratio)에서는 차이가 없으면서 L1D 캐쉬 미스 율은 줄어 듬을 확인할 수 있었다. 현재 32 비트 int 형으로 되어 있는 정수 데이터들을 살펴보고, 줄여도 영상 품질 성능에 별 문제가 되지 않는 데이터들을 short (16비트 등) 등으로 변환하는 실험을 수행 중에 있다.

IV. 실험 및 결과 검토

1. 실험 환경

본 논문의 실험에서 TMS320C6713 환경으로 TI 사의 6713DSK 를 사용하였다. 6713DSK는 평가보드 및 호스트에서의 통합 개발환경인 코드 컴포우저 스튜디오(Code Composer Studio, CCS)를 지원한다. CCS는 어셈블러, C 컴파일러, 링커, 프로파일러, 디버거 등을 지원한다. 평가보드의 6713 DSP는 225MHz로 동작하며, 외장 메모리로 사용되는 8MB의 SDRAM은 100MHz로 동작한다. 플래쉬 메모리 512KByte를 지원하며, 테스트를 위한 I/O를 지원한다. 본 논문의 실험에서 호스트는 평가보드 사이의 통신으로 XDS510USB라는 인서키트 에뮬레이터를 이용하여 JTAG 포트로 평가보드와 통신하였다. 호스트는 Pentium 4-1GHz, SDRAM 512MByte, 윈도우즈 XP 환경을 사용하였으며, CCS 버전은 V2.3을 이용하였다. 사용한 H.263 인코더는 TMN 8.0에 기반 한 UBC(University of British Columbia) H.263 version 2(H.263+) 구현[7]으로 C 소스 코드의 크기는 23,000 라인(720K바이트)이다. 실험에서는 베이스라인 모드만을 테스트하였고 선택한 영상 포맷은 QCIF이며, 테스트 영상은 테스트 영상으로 많이 쓰이는 foreman.qcif 이다. 움직임 벡터 추정에는 전 탐색(fast full search) 알고리즘을 선택하였으며, 역 DCT 는 빠른 IDCT(FastIDCT) 알고리즘을 사용하였다. 한편 컴파일러의 옵션은 -o2을 사용하였는데, 이 옵션은 소프트웨어 파이프라이닝을 지원한다.

수행 속도 개선 성능 평가는 H.263 인코더의 인트라 모드 압축 1회, 인터모드 압축 3회 의 총 4회를 수행하는 데 걸리는 CPU 클록수로 하였으며, CPU 클록 측정 실험은 TMS 320C6713 DSK를 사용하여 수행하였다.

한편, H.263 프로파일링 분석을 위해서는 TMS320C6713 Functional Simulator를 이용하였고, 캐쉬 히트율 분석 시뮬레이션 툴로는 TI 사에서 제공 하는 캐쉬 ATK(Analysis ToolKit)를 사용하였다.

2. TMS320C6713에서의 H.263 프로파일링

다음 [표 1]은 인트라모드 1 프레임 인코딩과 인터모드 1 프레임 인코딩을 수행했을 때, 호출 횟수가 많은 H.263 상위 5개 함수들과 호출횟수를 정리한 것이다.

표 1. H.263 함수들의 호출회수 프로파일링

함수명	SAD_Macroblock()	putbits()	put_coeff()	Quant_blk()	Dct()
호출회수	77,439	6,169	2,286	1,188	1,188

다음 [그림 2]는 인트라모드 1 프레임 인코딩과 인터모드 1 프레임 인코딩 프로파일링 결과, 총 수행 소요시간이 많은 상위 함수들의 총 수행 CPU 사이클 수에 대한 프로파일링 결과를 보여 준다.

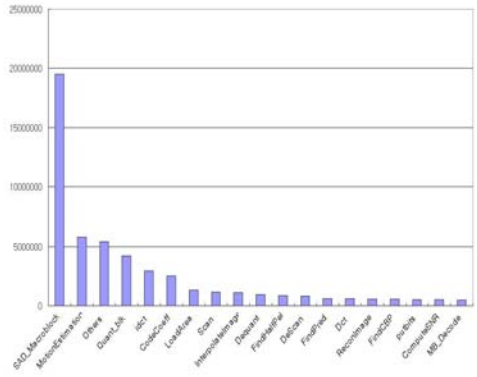

그림 2. 총 수행 CPU 사이클 수 프로파일링

3. L2 캐쉬 구성 분석

L2 캐쉬 구성에 의해 달라지는 L2 용량의 차이에 따른 성능을 살펴보기 위해 내부 메모리를 전혀 이용하지 않은 상태로 L2 캐쉬를 L2 캐쉬없음(No cache, 0KB), 직접사상 캐쉬(1 way; 16KB), 2중 집합연관 캐쉬(2

way; 32KB), 3중 집합연관 캐쉬(3 way; 48KB), 4중 집합연관 캐쉬(4 way; 64KB)로 각각 구성하고 각 경우에 대해 H.263 인코딩을 수행하고 소요된 CPU 클록 사이클 수를 각각 조사하였다.

표 2. L2 캐쉬용량에 따른 CPU 사이클 수

L2 캐쉬 용량	No cache	1 way	2 way	3 way	4 way
CPU 사이클 수	100,446,409	85,889,298	68,900,091	64,249,990	6,1071,130
CPU 사이클수 감소율(%)	기준	85.5	66	64	60.8

[표 2]는 이 실험 결과를 보여준다. [표 2]의 결과에서 보면, 2중 집합연관 캐쉬의 경우에 가장 많이 성능이 개선되었음 알 수 있으며, 3중 집합연관 캐쉬 이상부터는 L2 캐쉬 용량의 영향이 비교적 적음을 알 수 있다.

4. L2 내부 메모리 배치 분석

L2 캐쉬를 No cache, 2중, 4중 집합연관 캐쉬로 각각 구성한 경우, 늘어난 내부 메모리 64KB, 32KB, 0KB의 효과를 알아보기 위해 각 경우에 코드나 데이터 를 내부 메모리에 늘어난 만큼 더 배치하여 H.263 인코딩을 수행하여 소요된 CPU 클록 사이클 수를 조사하였다. 늘어난 내부 메모리를 포함하여 내부 메모리에 3.2절에서 설명한 대로 H.263 인코더를 프로파일링 하여 수행 시간이 많이 소요되고 자주 호출되는 루틴들을 구하고 이들 루틴들과 이 루틴들과 호출 관계에 있는 루틴들, 데이터 세그먼트, 스택 세그먼트 부분 등을 배치하였다. 스택 세그먼트는 12KB로 조정하였다.

No cache의 경우, 이 때 가능한 내부 메모리는 256KB이다. 실험에서는 이 내부 메모리에 데이터 세그먼트, 스택 세그먼트(12KB로 세팅), 코드 세그먼트의 루틴 중 소요시간 및 호출 횟수를 고려하여 최대한 많은 루틴을 배치하였다. 이 경우에 이용한 내부 메모리 사용량은 225KB이었다. 2중 집합연관 캐쉬의 경우, 가능한 내부 메모리는 224KB이다. 이 경우에 이용한 내부 메모리 사용량은 198KB이었다. 4중 집합연관 캐쉬의 경우, 가능한 내부 메모리는 192KB이다. 이 경우에 이용한 내부 메모리 사용량은 133KB이었다. [표 3]은 두 번째 실험의 결과이다.

표 3. L2 내부 메모리 배치에 따른 CPU 사이클 수

L2 캐쉬 용량	No cache	2 way	4 way
CPU 사이클 수	80,658,995	66,895,291	65,851,080

실험 결과 No cache와 2 way의 경우, 모두 내부 메모리를 사용하지 않은 경우에 비해 개선되었다. 그러나 4 way의 경우는 오히려 내부 메모리를 이용하지 않은 경우에 비해 더 떨어졌다. 이는 L1 캐쉬에서의 캐쉬 미스가 오히려 빈번하게 발생한 결과로 보인다. 즉, L2 내부 메모리에서 코드 및 데이터, 스택 세그먼트의 배치가 최적화 되지 않아서 이들 코드, 데이터, 스택 등이 외부 메모리에 배치될 때보다 L1 충돌 캐쉬 미스를 더 초래 한 것으로 판단된다.

이상의 실험을 통해서, 3.1절의 분석에서 살펴 본바와 같이 L2 캐쉬를 이용하는 경우가 하지 않는 경우보다는 성능이 더 나으므로 L2 캐쉬는 필요하고 또한 그 용량은 32KB 2중 집합연관 캐쉬가 적합 한 것임을 알 수 있다. 또한, 더 나은 성능 개선을 위해서는 L1 캐쉬 최적화가 필요함을 알 수 있다.

5. L1P 캐쉬 성능 개선 분석

다음 실험에서는 특별히 메모리 배치를 고려하지 않고 컴파일하고 링크하여 생성된 H.263 인코더 실행 파일(조정전)과 SAD_Macrobrock(), MotionEstimation(), Idct(), Quant_blk(), DCT(), CodeCoeff(), Load Area(), InterpolateImage(), DeQuant(), FindHalPel(), ReconImage(), MB_Decode(), Quantization() 등 자주 호출되며 수행시간이 많이 걸리는 루틴들과 이들 루틴과 호출관계가 있는 루틴들을 내부 메모리에 연속적으로 메모리 배치하도록 링커를 조절하여 생성된 H.263 인코더 실행 파일(조정후)에 대해 ATK 로 시뮬레이션하고 비교하였다. 이때, L2 캐쉬는 2중 집합연관 캐쉬로 구성하였다.

[표 4]는 이 실험 결과를 보여 준다.

표 4. L1P 캐쉬 미스 시뮬레이션 결과

	조정전(before)	조정후(after)
Read 히트 횟수	43,701,486	46,112,575
Read 미스 횟수	2,758,782	407,693
히트 율 (%)	94.06	99.12

시뮬레이션 결과, L1P 캐쉬의 히트 율이 5% 정도 개선되었음을 알 수 있다. 이러한 개선은 내부 메모리에서의 코드의 최적화 배치 작업에 의한 것이다. 실제로 각각의 실행 파일을 6713 DSK에서 수행하여 비교한 결과는 다음 [표 5]와 같다.

표 5. L1P 캐쉬 미스 실험 결과

	조정전(before)	조정후(after)
CPU 클록수	66,895,291	65,555,936
CPU 사이클 수 감소율 (%)	기준	98%

실제 실험에서 L1P 캐쉬 히트 율 개선에 의해 수행시간(CPU 클록수)은 2% 정도 개선이 이루어 졌음을 확인할 수 있다. 이러한 개선 효과는 인터모드 코딩 프레임 수가 증가 할수록 커진다.

6. L1D 캐쉬 성능 개선 분석

Idct 함수 경우에, 배정도(double precision)로 정의되어 있는 실수 데이터(조정전)를 단정도(single precision)바꾸고(조정후) 실험하였다. [표 6]은 실제로 6713 DSK에서 수행하여 비교한 결과를 보여 준다.

표 6. L1D 캐쉬 미스 실험 결과

	조정전(before)	조정후(after)
CPU 클록수	66,895,291	64,420,165
CPU 사이클 수 감소율 (%)	기준	96.3%

이때, 조정전은 [표 5]의 조정전과 동일한 상황이며, 조정 후는 조정 전과 같은 내부 메모리 배치에 단지 배정도 실수 데이터를 단정도로 바꾼 H.263 인코더를 수행한 결과이다. 조정 전과 조정 후의 PSNR 에는 차이가 없었음을 확인하였다. 수행시간이 3.7% 정도 개선되었음을 볼 수 있다. L1D에서 64 비트 데이터로 읽어온 후, 배정도 곱셈은 10 클록이 걸리지만, 단정도 곱셈은 4 클록이면 계산이 완료된다[4].

V. 결 론

본 논문에서는 TMS320C6713 에 H.263을 이식하는 경우에 있어서 고려하여야 할 캐쉬 및 내부 메모리 성능 분석에 대해 연구한 결과를 기술하였다. 분석 결과 L2 캐쉬는 2중 집합연관 캐쉬로 구성하고, 남는 용량은 내부 메모리로 이용하는 것이 효율적임을 확인하였다. 또한, TMS320C671x의 L1P 캐쉬는 직접사상 캐쉬이므로, 자주 사용되고 계산량이 많은 루틴을 중심으로 호출관계가 있는 루틴을 메모리에 연속으로 배치함으로써, 충돌 캐 미스를 줄이고 캐쉬 히트율을 높일 수 있음을 확인하였다. L1D 캐쉬의 경우, 간단한 데이터 타입 조정으로도 캐쉬 히트 율을 개선할 수 있음을 알 수 있었다. VLIW 구조에서의 성능 개선에는 소프트웨어 파이프라이닝에 기반을 둔 코드 최적화가 필요하다. 현재 소프트파이프라이닝 적용시의 캐쉬 동작 분석 및 캐쉬 성능 개선 방안에 대해 연구가 진행 중이다. 본 논문의 연구 결과는 고성능 DSP에 이식되는 S/W 의 최적화 방안에 도움을 줄 수 있을 것으로 기대된다.

참 고 문 헌

[1] 경종민, 박인철, 양진혁, 남상준, 이승종, 김병운, 박봉일, 박창재, 장유성, *고성능 마이크로프로세서 구조 및 설계 방법*, 대영사, 2000.

[2] P. R. Panda, N. D. Dutt, and A. Nicolau, "On-chip vs. off-chip memory: The data partitioning problem in embedded processor-based systems," ACM Trans. Design Automation Electron. Syst., Issue 3, Vol5, pp.682-704, 2000(7).

[3] P. R. Panda, "Data and Memory Optimization Techniques for Embedded Systems," ACM Transactions on Design Automation of Electronic Systems, pp.149-206, Vol.6, No.2, 2001(4).

[4] *TMS320C6000 CPU and Instruction Set Reference Guid,.* No. SPRU189, Texas Instruments, 2000(1).

[5] R. Cucchiara, M. Piccardi, and A. Prati, "Exploiting Cache in Multimedia," Proc. International Conference on Multimedia Computing and Systems (IEEE ICMCS99), Vol.1, Italy, pp.345-350, 1999(6).

[6] *TMS320C6713 Datasheet,* Texas Instruments

[7] B. Erol, F. Kossentini, and H. Alnuweiri, "Implementation of a fast H.263+ encoder/ decoder," Proc. IEEE Asilomar Conf. Alsilomar Conf. on Signals, Systems and Computers, Vol.1, pp.462-466, 1998(11).

[8] E. G. R. Iain, *Video Codec Design*, John Wiley, 2002.

[9] F. Jason and W. Wayne, "Multi-Level Cache Hierarchy Evaluation for Programmable Media Processors," IEEE Workshop on Signal Processing Systems, pp.59-85, 1998(3).

[10] F. Jason and W. Wayne, "Instruction fetch characteristics of media processing," SPIE Photonics West, Media Processors 2002, San Jose, CA, pp.72-83, 2002(1).

[11] F. Jason, W. Wayne, and L. Bede, "Understanding multimedia application characteristics for designing programmable media processors," SPIE Photonics West, Media Processors '99, San Jose, CA, pp.2-13, 1999(1).

[12] S. Bartolini, and C. A. Prete, "Optimizing instruction cache performance of embedded systems," ACM Transactions on Embedded Computing Systems (TECS), Issue4, Vol.4, 2005(11).

[13] P. R. Panda, N. D. Dutt, and A. Nicolau, "Memory Data Organization for Improved Cache Performance in Embedded Processor Applications," Proc. Int'l Symp. System-Level Synthesis (ISSS-96), pp.90-95, 1996(11).

[14] H. R. Sheikh, "Optimization of a Basline H.263 Video Encoder on the TMS320C6000," Proc. Texas Instruments DSP Educator's Conference, 2000(8).

[15] H. Miyazawa, H.263 Encoder: TMS320C6000 Implementation, Application Report SPRA721, Texas Instruments, 2000(12).

[16] *TMS320C6000 DSP Cache User's Guide.* No. SPRU656, Texas Instruments, 2000(1).

저 자 소 개

임 세 훈(SeHun Lim) 준회원

- 2007년 2월 : 숭실대학교 정보통신공학부 학사
- 2007년 3월 ~ 현재 : 숭실대학교 대학원 전자과 석사 과정 재학

<관심분야> : 임베디드 컴퓨팅

정 선 태(Sun-Tae Chung) 정회원

- 1990년 12월 : 미국 미시간대학교(앤아버) 전자 및 컴퓨터 박사
- 1991년 ~ 현재 : 숭실대학교 정보통신전자공학부 교수

<관심분야> : 임베디드 컴퓨팅, 생체인식, 컴퓨터 비젼, 영상 감시