TMS320C6201을 이용한 H.263 동영상 부호화기의 실시간 구현

김민성 , 정재호
인하대학교 전자공학과

Real-time Implementation of H.263 Encoder Using TMS320C6201

M.S.Kim , J.H.Chung
Dept. of Electronic Engr., Inha Univ.
e-mail : milenium007@hanmail.net , sunniechung@yahoo.com

요약

본 논문에서는 TI사의 TMS320C6201 DSP를 이용하여 H.263 동영상 부호화기를 실시간 구현하고자 한다. 구현한 부호화기는 QCIF 형식의 영상을 사용하여 ITU-T H.263 권고안의 기본 모드를 따라 주로 C 언어와 intrinsics를 사용하여 구현하였다. 특히, 속도 향상을 위해서 고속 메모리의 사용을 극대화하는데 중점을 두었고, 연산량이 많은 모듈에 대한 최적화와 데이터의 병렬 처리 및 DMA (Direct Memory Access) 전송 등을 고려하여 구현하였다.

1. 서론

최근에 많은 응용 분야에서 실시간 동영상에 관한 관심이 높아지고 있다. 예를 들면 화상 전화기, 화상 회의, 원격 차량 항법 시스템 등에 많이 이용되고 있다. 이러한 응용분야에서 실시간 처리를 위해서는 부호화 시간이 짧아야 하며 낮은 비트율의 출력 비트 스트림을 만들어야 한다. 일반적으로 실시간 동영상 부호화는 특별한 하드웨어에 의해 구현되는데 이는 알고리즘이 업그레이드 될 때 다시 설계해야하는 불편함이 있다. 하지만 이를 범용 DSP를 통해 소프트웨어적으로 구현함으로써 이런 불편함을 없앨 수 있다. 단지 구현시 문제가 되는 것은 DSP의 메모리가 제한되어 있으며 하드웨어로 구현할 경우보다 계산 능력이 떨어진다는 점이다. 본 논문에서는 이런 문제를 효과적으로 극복하여 실시간으로 구현할 수 있는 방법에 대해 제안하고자 한다. 2장에서는 사용된 DSP의 내부 구조에 대해 간략히 설명하고 3장에서는 구현된 동영상 부호화기의 형식 및 실시간 구현을 위한 전략을 설명한다. 4장은 실험에 사용된 시스템과 각 모듈별 측정 결과를 보여주고 마지막으로 5장에서는 실험에 대한 결론을 맺는다.

2. DSP의 내부 구조

TI사의 TMS320C6201 DSP는 200MHz에서 동작하고 두 개의 레지스터 파일 (A, B 측면)의 32개의 32비트 범용 레지스터와 2개의 곱셈기 (.M1, .M2), 6개의 산술기 (.L1, .L2, .S1, .S2, .D1, .D2)를 가지며 두 개의 레지스터 파일 크로스 패스 (1X, 2X)를 통해 A와 B 레지스터 파일들을 동시에 읽고 쓸 수 있다. .D 유닛에서 나온 데이터 어드레스 경로 .DA1, .DA2는 한 레지스터 파일에서 나온 데이터 어드레스로 다른 레지스터 파일로부터 메모리에 로드하고 저장할 수 있게 한다. 또한 클럭 사이클마다 패킷단위로 8개의 32비트 명령어까지 평행하게 수행할 수 있는 VLIW (Very Long Instruction Word) 구조로 되어있다. 모든 명령어는 조건적으로 수행하므로 거의 분기하지 않으며 병렬로 수행할 수 있다. 고속의 내부 메모리는 각각 64 KByte의 프로그램 메모리와 데이터 메모리로 구성되어있다. 프로그램 메모리는 CPU에 사이클 당 256비트의 패치 패킷을 제공할 수 있으며 데이터 메모리는 메모리 뱅크 히트가 일어나지 않는 한 매 사이클마다 병렬로 데이터를 액세스할 수 있다. 또한 저속의 외부 메모리는 256KByte의 SBSRAM과 16MByte의 SDRAM으로 구성되어있다. SBSRAM은 주로 코드 사이즈가 클 경우 프로그램 메모리로 사용되고 SDRAM은 전체 영상을 저장하는데 사용된다. 그 외에 프로그램 할 수 있는 4개의 DMA채널과 2개의 타이머가 있다.

3. 구현된 H.263 부호화기

구현된 동영상 부호화기는 H.263의 I-프레임과 P-프레임을 기본으로 구현되며 영상은 QCIF (176×144)형식을 사용한다. 부호화의 기본 단위는 매크로블록으로서 4개의 휘도 블록과 2개의 색차 블록으로 구성된 4 : 1 : 1 (Y : Cb : Cr) 영상 형식을 사용한다.

3.1 영상 압축 알고리즘

전체 알고리즘은 크게 화면내 코딩 (I- 프레임)과 화면간 코

딩 (P- 프레임)으로 구성된다. 두 코딩 모드의 블록도는 그림 1과 같다. 화면내 모드는 8×8 블록에 대해 DCT 및 양자화한 뒤 허프만 코딩을 수행한다. 한 편으로 양자화된 매크로블록을 다시 IDCT 및 역 양자화를 거쳐 원래의 매크로블록으로 복구하여 전체 영상을 복원한다. 화면간 모드는 이전 영상과 현재 영상으로부터 모션 추정 및 보상을 하고 여기서 보상된 차 매크로블록에 대해 DCT 및 양자화를 거쳐 허프만 코딩을 수행하고 한편 매크로 블록의 복구는 화면내 모드와 똑같이 IDCT 및 역 양자화에 의해 다시 차 매크로블록을 복원한 뒤 추정된 매크로 블록과 합하여 매크로블록을 복구하여 전체 영상을 복원한다.

3.2 구현 전략

3.2.1 메모리 사용의 최적화

외부 메모리는 대용량인데 반해 내부 메모리보다 30배 정도 속도가 느리며, 내부 메모리는 고속이지만 용량이 작은 단점이 있다. 하나의 매크로블록 (16×16)의 메모리간 데이터 이동 속도는 표 1과 같다. 이런 사실에 따라 속도 향상을 위해 적절한 메모리의 사용이 필요하다. 용량이 큰 입력 영상과 복구 영상은 외부 메모리에 저장하고 그 밖의 것들은 고속의 내부 메모리에서 처리하도록 한다. 입력 영상을 내부 메모리로 읽어 들일 때와, 영상을 복구 할 경우, 내부 메모리와 외부 메모리간의 데이터 이동이 있는데, 이때 DMA를 이용해 전송함으로서 CPU 연산의 지연 없이 데이터를 이동할 수 있다.

표 1. 메모리간의 데이터 이동 클럭 사이클 수

출발 / 목적	내 부	외 부
내 부	394	2147
외 부	4837	10575

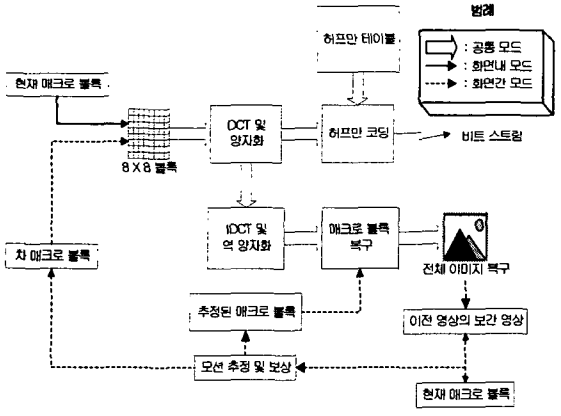
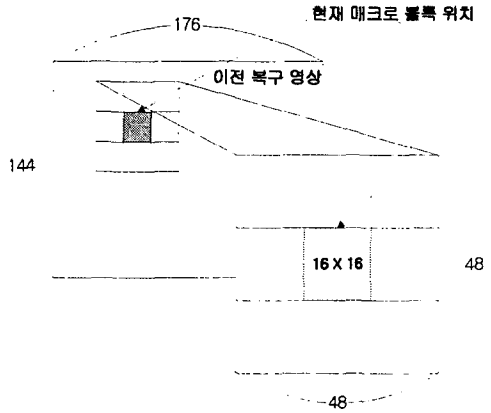


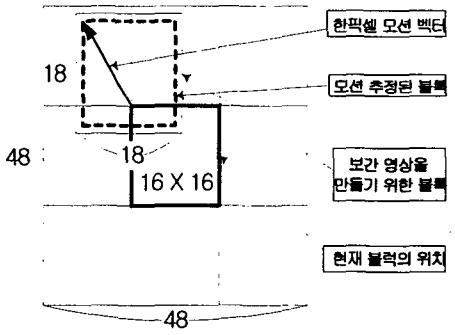
그림 1. 화면내, 화면간 코딩 모드의 블록도

3.2.2 모션 추정 알고리즘의 변형

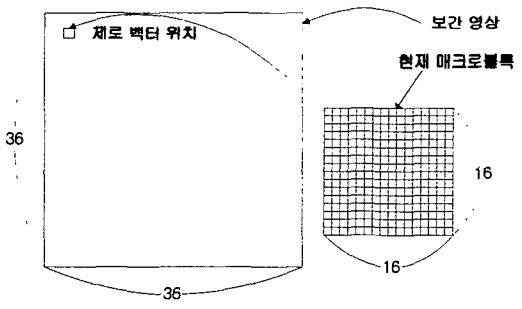
모션 추정을 위해서 이전 영상에 대한 보간 영상이 필요하다. 하지만, 보간 영상의 경우 외부 메모리에 있는 이전 영상에 대해 다시 외부 메모리에 보간을 하고 다시 반 픽셀 추정 시 내부 메모리로 읽어 와야 하므로 이에 소요되는 시간은 엄청나다. 그러므로 이것의 최적화를 위해 그림 2와 같이 알고리즘을 변형한다.



a) 한 픽셀 추정을 위한 48 × 48 블록



b) 보간을 위한 18 × 18 블록



c) 반 픽셀 추정을 위한 보간 영상

그림 2. 제안된 모션 추정을 위한 단계

그림 2의 a)는 복구 영상의 현재 매크로블록의 위치를 중심으로 주위의 8개의 매크로블록 (48×48)을 내부 메모리로 읽어 들인다. 그런 후, b)와 같이 추정 범위 내에서 SAD (Sum of Absolute Difference)가 최소가 되는 매크로블록을 찾아 한 픽셀 벡터를 구하고, 보간 영상을 만들기 위해 주위의 한 픽셀씩을 더 읽어들여 18×18 블록을 만든 후, c)와 같이 이에 대한 보간 영상 36×36 블록을 만들어 반 픽셀 추정을 한다. 그리고, 영상의 코너와 경계 부분은 특별한 경우로 처리한다. 이와 같은 알고리즘의 변형은 외부 메모리로부터 내부 메모리로의 데이터 이동이 단지 한 픽셀 추정을 위해 9개의 매크로블록을 읽어들일 때만 일어나고 나머지는 모두 내부 메모리에서 처리할 수 있도록 한다. 그러므로 모션 추정 및 모션 보상에 소요되는 시간이 기존의 알고리즘보다 약 3.5배 단축시킬 수 있다.

3.2.3 DMA를 이용한 핑퐁 전송

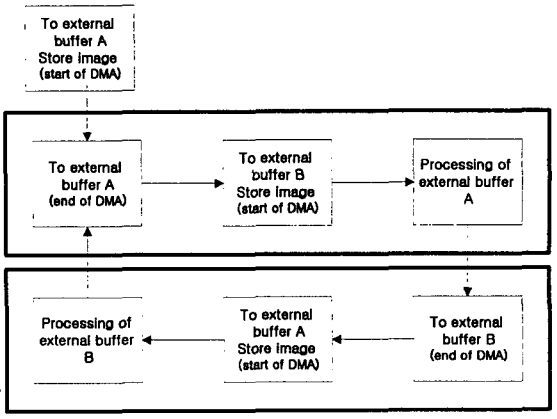
DMA전송은 CPU연산에 관계없이 메모리간에 데이터를 이동시킨다. 그러므로, 다음에 연산할 데이터를 현재 CPU가 연산하고 있는 동안 데이터를 이동시킴으로써 데이터 이동을 숨기고 CPU는 연산을 해 나갈 수 있다. DMA전송은 카메라로 획득된 입력 영상의 외부 메모리로의 전송과 내부 메모리로의 매크로블록 전송 및 비트 스트림의 전송, 복구된 매크로 블록의 외부 메모리로의 전송이 있다. 그림 3은 이에 대한 각각의 DMA전송의 블록도를 보여준다. 그림 3의 a)는 입력 영상이 외부 버퍼 A, B에 DMA 전송으로 번갈아 저장되고, 한 버퍼에 저장된 데이터가 처리되는 동안 기다림 없이 동시에 다른 버퍼에 DMA 전송을 시작한다. 이와 같이 현재 데이터를 처리하는 동안 다음에 처리될 데이터를 DMA전송을 통해 미리 로딩하는 것을 핑퐁 전송이라 한다. 그림 3의 b), c), d)도 이러한 핑퐁 전송을 기본으로 하여 최대한 데이터 전송에 의해 CPU 연산 지연을 없애고 계속 부호화를 할 수 있게 한다.

3.2.4 루프의 최적화

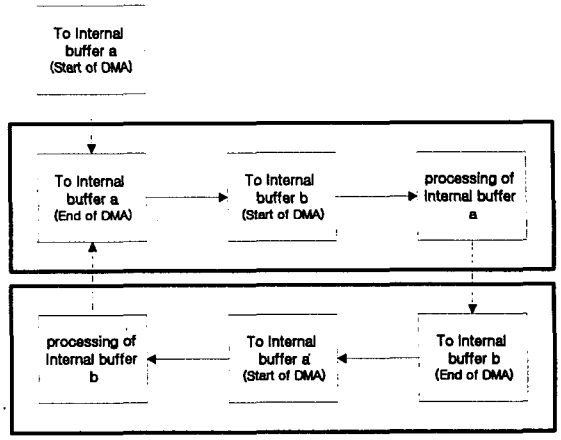
동영상의 부호화 과정은 대부분 반복적인 루프의 계산이 많다. 그러므로 이에 대한 최적화는 필수적이다. 'C6x 컴파일러가 제공하는 intrinsics은 어셈블리어로 직접 매핑되는 특별한 함수이다. 이를 이용해 16비트형 데이터를 32비트형으로 액세스하여 루프를 한번 수행할 때 두 개의 데이터를 처리할 수 있다. 또한 언롤링 기법을 이용해 분기를 최대한 없애고 한번 루프를 수행할 때 더 많은 데이터를 처리할 수 있다

3.2.5 내부 프로그램 메모리의 캐쉬 사용

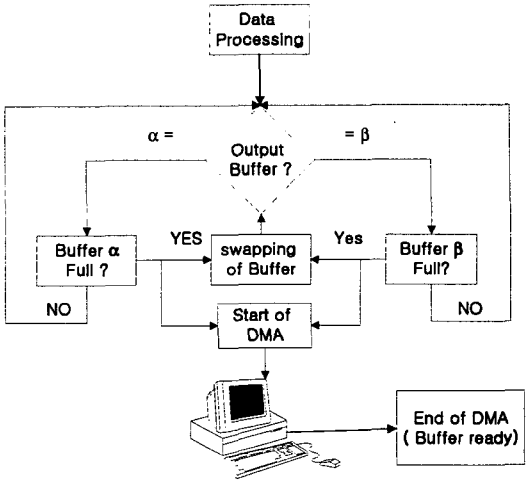
DSP 칩 안의 내부 프로그램 메모리는 DSP 코어 클럭과 동일하게 고속으로 동작하지만 용량이 작다. 아쉽게도 구현할 동영상 부호화 코드는 내부 메모리에 맞지 않는다. 하지만 내부 프로그램 메모리를 캐쉬로 사용하고 SBSRAM을 프로그램 메모리로 사용할 경우, 내부 프로그램 메모리를 사용하는 경우와 비슷한 성능을 가지므로 실시간 구현이 가능하다.



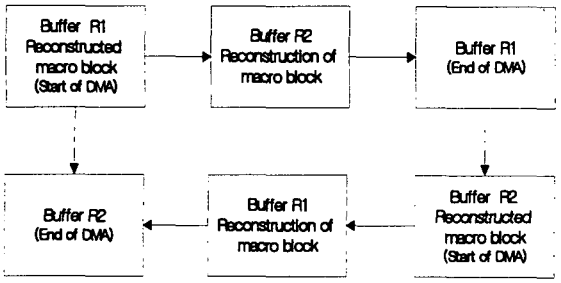
a) 외부 버퍼 A, B의 DMA 전송



b) 내부 버퍼 a, b의 DMA 전송



c) 출력 버퍼 α, β의 버퍼링



d) 복구된 매크로블록 버퍼 R1, R2의 DMA 전송

그림 3. DMA 채널의 사용 블록도

3.2.6 DCT 및 양자화의 최적화

DCT 및 IDCT는 TI사에서 제공하는 어셈블리 루틴을 사용하여 연산 속도를 개선하였다. 양자화의 경우 나눗셈 연산을 많이 사용하는데 나눗셈 유닛이 없으므로 덧셈 연산보다 20배 이상의 시간이 소요된다. 그러므로 나눗셈 연산을 곱셈과 쉬프트 연산으로 대치하여 속도를 향상시켰다.

4. 실험 및 결과

실험에 사용된 시스템은 그림 4와 같다.주로 C언어를 이용하여 프로그래밍 하였으며, 시뮬레이션용 보드는 Coreco사의 Cobra/C6을 사용하였다. 이것은 프레임 그래버 기능과 신호처리에 필요한 DSP를 가지고 있어서 다양한 신호처리 응용에 사용할 수 있다. 디지털 입력 영상은 CCD 카메라로 얻어진 아날로그 영상을 프레임 그래버를 통해 얻어지고, 이 영상은 DMA 전송을 통해 DSP에 전송되어 부호화되며 압축된 데이터는 PCI를 통해 호스트(PC)에 저장된다. 표 2는 CCS (Code Composer Studio)의 프로파일 기능을 이용하여 최적화 이전과 이후의 사이클 수를 비교한 것이다. 하나의 매크로블록을 기준으로 측정하였으며 177 - 210 사이클의 오버헤드를 포함한다. 특히, 메모리 사용을 최적화하고 모션 추정 알고리즘의 변형으로 인해 약 70%의 속도 향상을 보였고 DCT 및 양자화의 최적화로 인해 약 92.7%의 속도 향상을 보였다. 전체적으로 구현된 동영상 부호화기는 초당 약 15프레임을 처리할 수 있는 성능을 보였다.

5. 결론

본 논문에서는 TMS320C6201을 이용하여 H.263 동영상 부호화기를 실시간으로 구현하기 위하여 여러 가지 방법을 제안하였다. 특히, 속도 향상을 위해 고속 메모리의 사용을 극대화하고, DMA전송을 이용해 연산 지연을 없앴으며, 모듈별로 최적화하였다. 향후 더 나은 성능 향상을 위해서 보간 영상을 만드는 루틴, 전역 탐색 루틴과 SAD루틴과 같이 상대적으로 시간이 많이 소요되는 루틴에 대해 완전히 어셈블리어로 프로그래밍 하고자 한다.

표 2. 최적화 이전과 이후의 모듈별 측정된 사이클 수
(TI 컴파일러의 -o3 옵션 사용)

함 수	최적화 이전	최적화 이후
DCT	17190	1218
IDCT	19201	1404
양자화	37440	1326
모션 추정	420599.7	117956.2
모션 보상	33728.4	13377.7
보간 영상	93626.4	14620
반 픽셀 추정	78403.4	37007.2

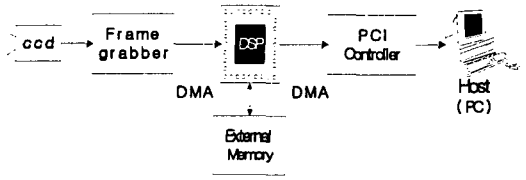


그림 4. 실험 시스템의 개략도

참고 문헌

[1] Draft Text of Recommendation H.263 Version 2 ("H.263+") for Decision,International Telecommunication Union, 1997 - 2000

[2] C60 Native API Software Reference Manual, Coreco Inc., 1999

[3] TMS320C62x/67x Programmer's Guide, Texas Instruments, 1999

[4] Hamid R. Sheikh, Serene Banerjee, Brian L. Evans, and Alan C. Bovik, "Optimization of a baseline H.263 Video Encoder on the TMS320C6000", Laboratory for Image and Video Enginnering Dept. of Texas at Austin

[5] S. Sriram and C. Y. Hung, "MPEG-2 Video Decoding on the TMS320C6x DSP Architecture," in proc. IEEE Asilomar Conf. On Signals, Systems and Comp., vol. 2, pp. 1735-1739, Nov. 1998.