TMS320C6201을 이용한 DVR 시스템을 위한 영상 부호화기 구현

최용석*, 금재혁**, 임중곤*, 민홍기***, 박종승**, 정재호*
*인하대학교 전자공학과, **주식회사 아임스, ***인천대학교 정보통신공학과

Real-time Implementation of Image Encoder for DVR Systems using TMS320C6201

Y.S.Choi*, J.H.Keum**, Z.K.Yim*, H.K.Min***, J.S.Park**, J.H.Chung*
*Dept. of Electronic Engr., Inha Univ.
**IMS Inc., Korea
***Dept. of Info. & Telecom. Engr., Univ. of Inchon
E-mail : g1991189@inhavision.inha.ac.kr

요약

본 논문에서는 TMS320C6201 DSP (Digial Signal Processor)를 이용하여 실시간 영상 부호화기를 구현하였다. 기본적인 영상 압축 방법으로는 baseline-JPEG을 사용하였고 이에 움직임 검출 알고리즘을 부가하여 영상의 시간적인 중복성을 제거하였다. 특히 저속 메모리와 고속 메모리의 효율적인 분배 사용, 계산량이 많은 모듈의 최적화, 데이터의 병렬 연산과 DMA (Direct Memory Access)를 이용한 데이터 전송 등의 방법을 통하여 실시간 영상 부호화기의 고속 영상 처리에 중점을 두었다.

1. 서론

최근에 영상 분야에서 디지털화가 빠르게 진행되고 있다. DVR (Digital Video Recoding)은 영상을 디지털로 저장하는 것으로서, 아날로그 방식에 비해 검색이 용이하고 화질 열화가 없는 장점이 있다. DVR은 응용에 따라서 그 특성에 차이가 있는데, 본 논문에서는 감시 카메라 등 보안 분야에 사용되는 DVR에 대해 연구하였다. 보안 분야의 응용에서는 영상의 입력이 다채널인 경우가 많고, 영상의 연속적인 재생보다는 정지 영상의 화질이 중요하다. 한편 영상을 디지털로 저장할 경우 매우 큰 저장공간이 요구되므로 영상의 압축이 필수적이고, 압축 과정은 계산량이 매우 크기 때문에 압축 작업을 전담할 프로세서가 있는 것이 바람직하다.

본 연구에서는 영상의 압축을 위해 Texas Instruments사의 TMS320C6201 DSP를 사용하였다. TMS320C6201 DSP는 200MHz의 속도로 동작하고 다채널의 DMA를 내장하고 있으므로 많은 양의 데이터를 처리할 수 있다. 또한 파이프라인을 지원하는데, 즉 2개의 곱셈기와 6개의 ALU (Arithmetic Logic Unit)를 통해서 8개의 명령을 동일 클럭에 병렬 처리할 수 있으므로 반복적인 수식 연산을 고속으로 수행할 수 있다.

2. 영상압축 알고리즘

영상의 압축을 위해 기본적으로 JPEG을 사용하였다. JPEG은 공간적인 영상의 중복성을 줄일 뿐이므로 움직임 검출 방법을 통하여 영상의 시간적인 중복성을 제거하였다. 동영상 부호화 방식을 이용하지 않은 이유는 여러 채널의 입력을 고려하였기 때문이다. 서로 다른 여러 채널의 영상을 MPEG등의 동영상 방식으로 압축하기에는 연산량의 측면에서나 요구되는 메모리 공간 등에서 무리가 있다. 본 연구에서 사용된 움직임 검출 방법은 연산량의 최소화를 우선적으로 고려하였는데, 움직임이 많거나 카메라 자체가 움직일 경우는 압축 효과가 적다. 다행히 보안 분야의 응용에서는 이런 경우는 거의 없으므로 간단한 움직임 검출 방법으로도 큰 효과를 볼 수 있다.

JPEG은 영상을 일정한 크기로 잘라서 부호화를 수행하는데 영상 부호화의 기본 단위는 MCU (Minimum Coded Unit)이다 [1]. MCU는 16×16의 화면 크기로 하였고 다시 6개의 블록으로 되어 있는데, 4개의 명도 (Y 신호) 블록과 2개의 색차 (Cb, Cr신호) 블록으로 구성된다. 각 블록은 8×8의 크기이다. 이와 같은 컬러 구성을 4:1:1이라고 한다 (Y:Cb:Cr=4:1:1).

각 8×8 블록에 대해서 그림 1과 같은 압축 과정을 거친다. DCT를 통해서 에너지의 집중이 일어나고, 양자화를 통해서 시각적으로 둔감한 부분의 정보가 제거되며, 허프만 부호화를 거치면서 부호 자체의 중복성이 제거된다. 한 개의 MCU에 대해서 그림 1의 과정이 6번 일어난다.

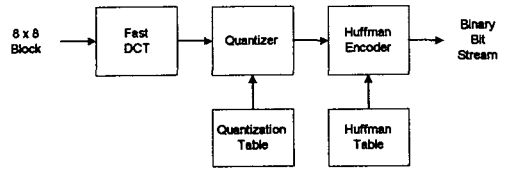

그림 1. 한 개의 8 × 8 블록의 부호화

2.1 고속 DCT 알고리즘 사용

DCT는 매우 큰 연산량을 가지고 있지만 동일한 연산의 반복으로 구성되어 있기 때문에 TMS320C6201의 병렬처리 기능을 이용하면 큰 효과를 볼 수 있다. 고속의 DCT에 대해서는 많은 연구가 수행되어 있는데, 본 논문에서는 *아라이, 아구이, 나카지마*가 제안한 DCT (이하 AAN DCT라고 하겠습)를 사용하였다 [2]. AAN DCT는 버터플라이 계산법과 행렬의 분해를 통해서 계산량을 현저히 감소시키고 있다. 결과적으로 8개 화소에 대한 1차원 DCT는 5번의 곱셈과 29번의 덧셈만으로 이루어진다. 2차원 DCT의 경우 먼저 1차원 DCT를 행방향으로 계산하고 이 결과에 다시 열방향으로 1차원 DCT를 수행한다. 그림 2에 1차원 AAN DCT의 계산과정을 도시하였다. 그림에서 선 위에 쓰여진 숫자는 곱셈 계수이고, 검은색으로 채워진 원은 덧셈을 뜻한다.

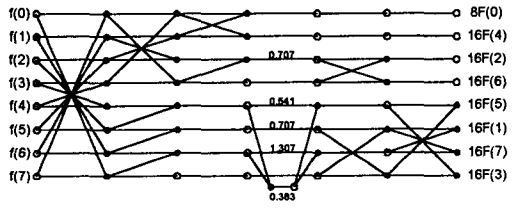

그림 2. 1차원 AAN DCT의 계산과정

2.2 양자화 과정의 개선

양자화 과정에서는 DCT 변환된 영상을 양자화 테이블의 값으로 나눗셈을 하게 되어있다. TMS320C6201은 나눗셈 유닛이 없으므로 에뮬레이션을 통해 처리하는데 이 연산은 덧셈보다 20배 이상의 시간이 소요된다. 따라서 이의 개선이 필수적이다. 양자화 역시 동일한 연산이 반복되므로 병렬처리를 사용했을 때 큰 효과를 볼 수 있는 부분이다.

본 연구에서는 나눗셈을 곱셈과 쉬프트 연산으로 바꾸어 계산하는 방법을 사용함으로써 빠른 동작을 가능하게 하였다 [4]. 정수 곱셈 연산으로의 변환을 위하여 JPEG의 표준 양자화 테이블을 그림 3과 같이 변형하였다. 변형된 양자화 테이블의 값은 다음과 같이 구했다.

$$modified\_Qtable = 2^{18} / Qtable$$

여기에서 *Qtable*은 그림 3.(a)의 각 값을 가리키고, *modified_Qtable*은 그림 3.(b)의 각 값을 나타낸다. 수식에서 2^{18} 이란 값을 사용한 이유는 그림 3.(b)의 값이 16비트 변수형이 허용하는 범위 내에서 최대값을 갖게 하여 정수 연산에 의한 오차를 최소화하기 위한 것이다. 그림 3.(b)에서 가장 큰 값은 26214로서 16비트 변수의 범위인 -32768~32767 내에서 최대로 스케일링되었다.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

(a)

16384	23831	26214	16384	10923	6554	5140	4297
21845	21845	18725	13797	10082	4520	4369	4766
18725	20165	16384	10923	6554	4599	3799	4681
18725	15420	11916	9039	5140	3013	3277	4228
14564	11916	7085	4681	3855	2405	2545	3404
10923	7490	4766	4096	3236	2521	2320	2849
5350	4096	3361	3013	2545	2166	2185	2595
3641	2849	2759	2675	2341	2621	2545	2648

(b)

그림 3. (a)JPEG의 표준 양자화 테이블,
(b)변형된 양자화 테이블

2.3 움직임 검출

한 화면내의 영상의 중복성에 대해서는 DCT와 양자화를 통해서 줄일 수 있었다. 이외에도 이전 영상과 현재 영상 사이에는 많은 중복성이 있다. 즉 시간적인 중복성이 있는데, 만약 영상간에 변화 부분만을 저장할 경우 압축률을 크게 높일 수 있다. 따라서 움직임 검출 방법을 사용하였는데 이는 움직임이 일어난 부분은 새로 부호화하고 움직임이 일어나지 않은 부분은 이전 영상을 이용하는 방법이다.

움직임 검출을 위해 우선 한 장의 화면이 기준 영상이 되고 이는 이후 일정 개수의 화면과 비교된다. 움직임 검출의 기본 단위는 앞서 언급한 MCU로 하였다.

MCU 내에서 색차 블록을 제외한 명도 블록만이 움직임 검출에 이용되는데, 이는 영상 내의 대부분의 정보가 명도 성분에 몰려있다는 시각 특성에 근거한 것이다.

움직임 검출의 과정을 보면, 현재 영상의 MCU와 기준 영상의 MCU가 서로 비교된다. 만약 움직임이 검출되면 MCU의 위치가 움직임 테이블에 기록되고 이후 DCT, 양자화, 허프만 부호화를 거친다. 한편 움직임이 없는 MCU에 대해서는 아무런 일도 일어나지 않는다. 결과적으로 부호화된 비트 스트림에는 움직임이 일어난 MCU의 데이터들만이 차례대로 기록되어 있다. 복호화 시에는 움직임 테이블의 위치 정보를 참조하여 해당 위치에 MCU를 복호화하고, 움직임 테이블에 기록되어 있지 않은 MCU에 대해서는 기준 영상의 데이터를 이용한다. 한편 기준 영상의 부호화에는 움직임 검출을 사용하지 않고 모든 MCU가 부호화된다.

움직임 검출 방법으로는 SAD (Sum of Absolute Difference)를 사용하였는데 다음 식과 같이 정의된다.

$$SUM = \sum_{i}^{16}\sum_{j}^{16} |reference_{(i,j)} - current_{(i,j)}|$$

즉, 현재 영상의 MCU와 기준 영상의 MCU를 화소 단위로 차 신호를 구하여 이 값을 전부 합한 후, 미리 설정해 놓은 기준값(Threshold)과 크기를 비교한다. 비교되는 값이 기준값보다 클 경우에는 움직임이 일어났음을 의미한다.

2.4 DMA를 이용한 핑퐁 전송

데이터의 처리 시간을 단축시키고, 메모리 간의 속도 차이를 극복하고자 핑퐁 전송을 사용하였다 [5]. 고속의 데이터 처리를 위해서는 빠른 속도의 메모리를 사용해야 하는데 고속의 메모리는 그 용량이 적기 때문에 효율적으로 활용하여야 한다. 따라서 현재 처리할 데이터만을 고속 메모리에 올려놓고 사용하게 되는데, 이 때문에 빈번한 메모리 전송이 발생한다.

핑퐁 전송의 기본은 현재 데이터를 처리하는 동안 다음에 처리될 데이터를 미리 로딩하는 것이다. 이 동작은 DMA를 통해서 가능하다. 원래는 한 개의 블록이 메모리에 로딩되고 이 데이터에 대한 처리가 수행되고, 다시 다음 블록이 로딩되는 등 순차적으로 부호화 과정이 진행된다. 하지만 DMA를 통한 데이터의 전송은 DSP의 계산과 독립적으로 일어나므로 데이터의 처리와 전송을 동시에 수행하는 것이 가능하다.

예를 들어 A와 B의 두 개의 버퍼가 있는 경우, A에 있는 데이터가 처리되는 동안 DMA를 이용하여 백그라운드 작업으로 B에 채우고, 이어서 B에 있는 데이터를 처리하면서 백그라운드 작업으로 A를 채운다. A와 B가 역할을 바꿔가면서 위의 동작이 연속적으로 반복된다. 본 연구에서는 두 부분에서 핑퐁 전송을 사용하고 있다. 그림 4에 두 가지 핑퐁 전송에 대해 나타내었다.

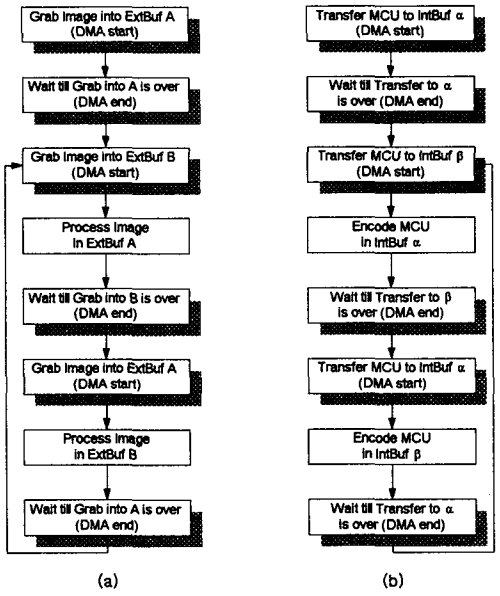

그림 4. 핑퐁 전송 (a)외부 메모리에서 일어나는 핑퐁 전송 (b)내부 메모리에서 일어나는 핑퐁 전송

3. 영상 부호화기의 구조와 동작

영상 부호화를 위한 시스템은 DSP와 그래버, 카메라, 호스트로 구성된다. 시뮬레이션용 DSP보드로 Coreco사의 Cobra/C6을 사용하였는데, 이 보드에는 아날로그 입력 영상을 디지털로 변환하여 버퍼에 저장하는 그래버 기능이 있고, TMS320C6201 DSP가 내장되어 있으므로 다양한 신호 처리 기능을 수행할 수 있다. 영상 입력을 위해 CCD 카메라를 사용하였고 호스트로는 펜티엄II PC를 사용하였다. 전체적인 시스템의 구조를 그림 5에 나타내었다.

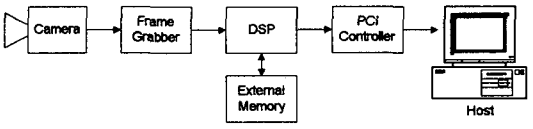

그림 5. 영상 부호화기의 구조

그림 6에 부호화 과정의 블록 다이어그램을 나타내었다. 동작 과정을 살펴보면, 카메라를 통해서 입력된 영상은 외부 메모리에 저장된다. 저장된 영상은 MCU 단위로 쪼개어지고 한 개의 MCU씩 고속의 내부 메모

리로 전송된다. 각 MCU에 대해 움직임 검출이 수행되는데, 움직임이 있는 MCU는 압축과정을 거친 후 출력버퍼에 축적되며, 움직임이 없는 MCU는 압축과정을 거치지 않는다. 출력버퍼가 꽉 차면 호스트로 데이터가 전송되고 한 화면의 부호화가 끝날 때까지 위의 과정을 반복한다. 이후 화면에 대해서도 같은 동작이 계속된다.

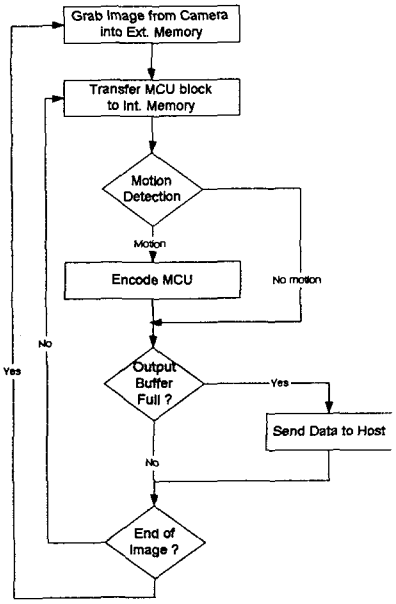

그림 6. 부호화 과정의 블록 다이어그램

4. 실험 결과 및 고찰

TMS320C6x를 위한 통합 툴인 코드 콤포저 스튜디오의 프로파일 기능을 이용하여, DCT와 양자화의 속도를 측정하였다. 각 값은 200MHz의 TMS320C6201 DSP칩에서 전용 컴파일러를 사용하여 생성된 실행파일을 구동하여 얻어진 결과들이다.

표 1은 수식을 그대로 구현한 경우와 AAN DCT를 사용했을 때의 속도 차이를 나타내고 있다. AAN DCT를 사용했을 때 약 10배의 속도 향상을 보임을 알 수 있다. 표 2는 양자화 개선에 의한 속도 향상을 보여준다. 나눗셈을 이용한 일반적인 양자화에 비해 곱셈을 이용하도록 변형된 양자화가 20배 정도 빠른 속도로 수행되고 있다. 여기에서 나눗셈 연산장치가 없는 기기에서 나눗셈을 수행할 경우 상당한 부하가 걸림을 확인할 수 있다.

전체적인 성능을 보면 한 개의 8×8 크기의 블록에 대해 3000~3500 싸이클이 소요되고, 한 개의 MCU에 대해서는 20,000 싸이클 내외의 부하가 걸린다. 본 연구에서 사용된 시스템은 CCD 카메라를 통한 320×240 크기의 24비트 컬러 입력에 대해 초당 30장 정도를 실시간으로 부호화할 수 있다.

표 1. AAN DCT에 의한 속도 향상
(TI 컴파일러에서 -o3 옵션 사용)

DCT 방법	straight forward DCT	AAN DCT
Clock Cycles	2561	273

표 2. 개선된 양자화에 의한 속도 향상
(TI 컴파일러에서 -o3 옵션 사용)

양자화	일반적인 방법	개선된 방법
Clock Cycles	4860	221

참고문헌

[1] International Standard of *Digital Compression and Coding of Continuous-tone Still Images, JPEG,* International Telecommunication Union, 1993.

[2] Y. Arai, T. Agui, M. Nakajima, "A Fast DCT-SQ Scheme for Images," *Trans. of the IEICE*, E-71, pp. 1095-1097, 1988.

[3] W.B. Pennebaker and J.L. Mitchell, *JPEG Still Image Data Compression Standard,* Van Nostrand Reinhold, 1993.

[4] 조경석, 고윤호, 김성대, "TMS320C62X를 이용한 H.263 부호화기 구현 및 동영상 처리를 위한 DSP 구조 연구," *신호처리 합동 학술대회 논문집*, vol. 12, no. 1, pp. 949-952, 1999.

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

[6] *TMS320C62x/67x Programmer's Guide,* Texas Instuments, 1999.

[7] *TMS320C6000 Code Composer Studio Tutorial,* Texas Instuments, 1999.

[8] "IJG JPEG software 6.0b," Independent JPEG Group, 1998.

본 연구는 인천대학교의 1999년도 RRC 프로그램과 IMS Inc.의 연구비 지원에 의하여 이루어 졌습니다.