H.264/AVC 디코더용 움직임 보상 연구

송형돈 · 손승일

한신대학교 정보통신학과

A Study on Motion Compensation for H.264/AVC Decoder

Hyeong Don Song, Seung Il Sonh

Dept. of Information and Communication Hanshin University

e-mail : brothermoney@hs.ac.kr

요 약

H.264/AVC는 다양한 블록 사이즈에 따라 움직임 보상을 수행한다. 본 논문은 1/4정밀도 화소를 지원하는 효율적인 움직임 보상에 대해 연구하였다. 참조 프레임의 데이터로 사용하기 위한 메모리의 접근을 줄이고 2개의 6-tap 필터를 사용하는 움직임 보상을 제안한다. 소프트웨어 검증을 통한 최적화된 알고리즘을 사용하여 하드웨어 설계 언어를 이용하여 기술하고 ModeSim 6.0a를 이용한 데이터 검증을 수행하였다.

키워드

움직임 보상, 보간, H.264/AVC

Ⅰ. 서 론

최근 영상을 중심으로 여러 형태의 정보를 결합하여 저장하거나 전송하는 멀티미디어가 많은 관심을 받고 있으며 제한된 채널과 데이터양 그리고 처리속도에 대한 문제점들이 제기되어 영상 압축 기술에 대한 관심이 높아지고 있다.

H.264/AVC는 ISO/IEC 의 MPEG와 ITU-T의 VCEG 두 그룹이 공동 연구기관 JVT를 창설하여 새롭게 제안한 동영상 압축에 관한 국제 표준이다[1].

H.264/AVC는 정수단위 DCT와 I 프레임에 대한 인트라 예측과 P, B 프레임에 대한 가변 블록 움직임 예측과 1/4 화소 단위 움직임 벡터 추정 기법을 사용한다. 이는 기존의 H.263이나 MPEG-4 SP(Simple Profile)에 비해 동일한 화질에 대하여 압축률이 1.5~2 배 더 좋다고 밝혀졌으나 복잡도는 MPEG-4 SP 보다 약 4배 높은 것으로 보고되었다[2][3].

부호화 기술들은 복잡도는 고려하지 않고 압축 효율에 중점을 두고 개발하였다.

H.264/AVC는 16x16, 16x8, 8x16, 8x8, 8x4, 4x8, 4x4 와 같은 7가지 다양한 블록사이즈를 사용하여 움직임 예측을 수행하기 때문에 디코더의 움직임 보상의 경우 예측된 블록사이즈에 따라 움직임 보상을 수행해야 한다. 이는 하드웨어 구현 시 칩 면적에 대하여 문제점을 야기한다[4].

이에 본 논문은 다양한 블록 사이즈에 대하여 효율적인 처리가 가능한 움직임 보상에 대하여 연구하였다.

모든 블록 사이즈를 지원하고 참조 프레임의 데이터로 사용하기 위한 메모리의 접근을 줄이며 1/4 화소를 처리하는 6-Tap 필터를 2개 사용하는 움직임 보상을 제안한다. 소프트웨어를 사용하여 알고리즘에 대한 검증을 하고 최적화된 알고리즘을 사용하여 하드웨어 설계언어를 이용하여 코딩한 후 ModelSim 6.0a를 이용한 데이터 검증을 통하여 확인하는 것으로 본 논문에 대한 결론을 내리고자 한다.

Ⅱ. 움직임 보상 연구

2.1 트리구조 움직임 보상

매크로 블록(16x16)은 그림 1과 같이 4가지 방법으로 나누어 질 수 있다.

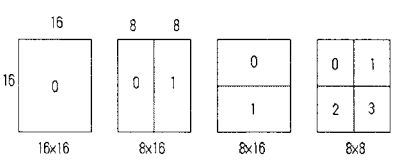

그림 1. 매크로 블록 파티션

하나의 16x16 매크로블록 파티션, 두 개의 16x8 파티션, 두 개의 8x16 파티션 또는 네 개의 8x8 파티션으로 움직임 보상 될 수 있다.

8x8 모드가 선택되면 매크로 블럭 내의 네 개의 8x8 서브매크로 블록은 다시 하나의 8x8 매크로블록 파티션, 두 개의 8x4 파티션, 두 개의 4x8 파티션 또는 네 개의 4x4 파티션으로 움직임 보상 될 수 있다.

이는 움직임 보상을 수행될 때 많은 수의 조합이 가능하다는 것을 보여준다[2].

2.2 움직임 보상 알고리즘

움직임 보상은 항상 하나의 매크로 블록 단위로 처리된다. 인코더에서 움직임 예측에 의해 계산된 움직임 벡터값을 사용하여 참조 프레임 데이터로부터 움직임 보상 데이터는 재구성된다. 움직임 보상 예측을 할 때 정수화소 이하의 화소 정밀도 신호는 참조픽처의 화소값들 간의 보간을 통해 생성된다[2][5].

그림 2는 보간 후 신호와 보간 전의 정수화소 신호와의 위치 관계를 보여주고 있다.

H.264/AVC는 화질 개선을 위해 1/2 화소와 1/4 화소를 사용한다,

알파벳 대문자는 정수화소를 나타내고 알파벳 소문자는 1/2 화소 및 1/4화소를 의미한다. 1/4픽셀의 보간은 움직임 추정과, 움직임 보상 그리고 프레임 복원과정의 복잡도를 증가시키지만, 1/2픽셀 움직임 보상과 비교하여 더 적은 오차값을 가진다.

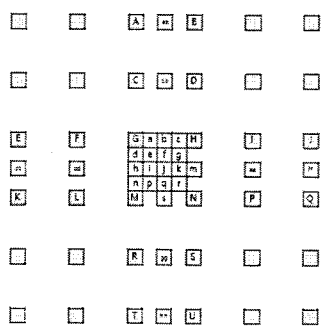

그림 2. 보간에 필요한 픽셀의 위치

2.2.1. 1/2 화소 생성

정수화소로부터 6-tap 필터와 가중치를 사용하여 정수 위치 샘플로부터 1/2 화소를 생성한다. 예로 수평 방향(E, F, G, H, I, J)에 대하여 6-tap 필터를 적용하면 아래 (식) 1,2 와 같다

$$b_1 = (E - 5F + 20G + 20H - 5I + J) \quad (1)$$

$$b = Clip((b_1 + 16) >>5) \quad (2)$$

수직 방향 1/2화소도 위의 식이 사용된다.

2.2.2. 1/4 화소 생성

1/2 화소 위치를 중심으로 2-tap 필터를 사용하여 1/4 화소를 생성한다.

화소 a, c, I, k는 주변에 인접한 정수화소 신호 또는 1/2화소 신호의 수평방향 평균치 필터를 사용하여 생성한다. 예를 들어 화소 a,f,r 은 아래 (식)3,4,5 에 의해 계산된다.

$$a = (G + b + 1) >> 1 \quad (3)$$

$$f = (b + j + 1) >> 1 \quad (4)$$

$$r = (m + s + 1) >> 1 \quad (5)$$

III. 제안하는 구조

본 논문은 다양한 블록 사이즈에 대한 움직임 보상을 가장 작은 블록인 4x4 블록 단위로 처리를 기본단위로 나누어 처리를 한다. 이로 인하여 4x4 블록이 아닌 블록에 대하여 4x4로 나누어 주는 모듈이 추가적으로 필요하지만 이는 다양한 블록 사이즈에 대한 움직임 보상을 구현하는 것보다 칩 면적을 적게 차지한다.

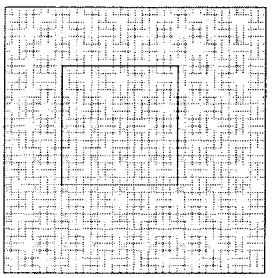

그림 3. 4x4 단위 보간에 필요한 주변 화소

그림 3은 4x4 단위로 움직임 보상을 수행하기 위해 필요한 9x9의 참조 화소들을 나타낸다. 회색으로 채워진 위치는 정소화소의 위치를 나타내며 나머지 흰색으로 채워진 곳은 1/2 화소 및 1/4화소의 위치를 보여준다.

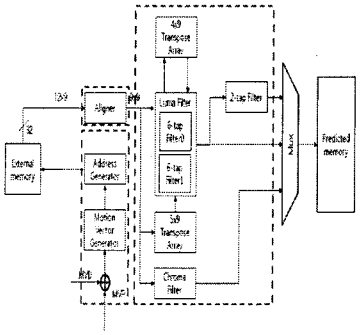

그림 4. 제안한 움직임 보상 전체 블록도

그림 4는 제안하는 움직임 보상 수행에 대한 전체 블록도이다.

4x4 블록으로 처리하기 위해 9x9화소를 메모리로 데이터를 읽어 오기 위해서는 한 라인의 12화소를 읽어 온 후 그림 4처럼 정렬기를 사용하여 그 중 필요한 9화소만 6-tap 필터로 전달한다. 4x4블록의 상단 좌표 x 값에 따라 12화소 중 9화소가 결정된다. 그림 5는 정렬기의 구조를 보여주고 있다.

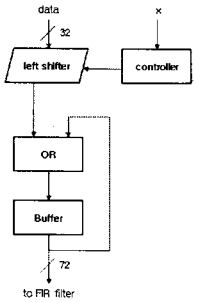

그림 5. 정렬기의 구조.

아래 그림 6은 정렬기의 내부 쉬프터의 연산 과정을 보여주고 있다.

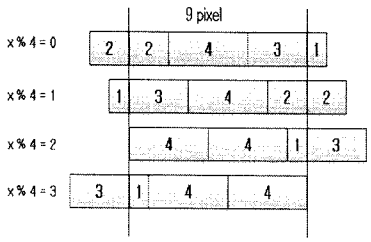

그림 6. 정렬기의 쉬프터 연산

9개의 화소는 2개의 6-tap 필터를 2번 사용하여 4개의 1/2화소를 구한다. 구해진 4개의 1/2화소는 그림 1의 f, q 화소와 같은 수직방향 1/4 화소와 j와 같은 1/2 화소를 를 구할 때 사용하기 위해 4x9 전치 배열과 레지스터로 전달된다. 그림 7은 4x9 전치 배열의 구조를 보여준다.

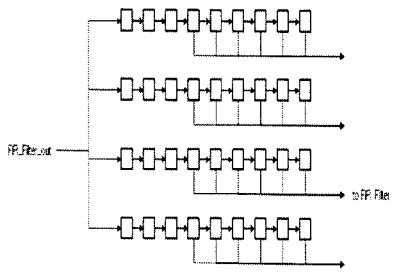

그림 7. 4x9 전치 배열의 구조

6-tap 필터를 2개 사용하기 때문에 하나의 행에 2사이클이 소요되고 전체 9행에 대해서는 29사이클이 소요된다. 또한 초기 입력된 정수 화소는 6-tap 필터의 입력과 동시에 그림 1의 h, m 화소를 수직 방향 1/2 화소를 구하기 위해 5x9 전치 배열 모듈에 입력되는데 모듈의 구조는 아래 그림 8과 같다.

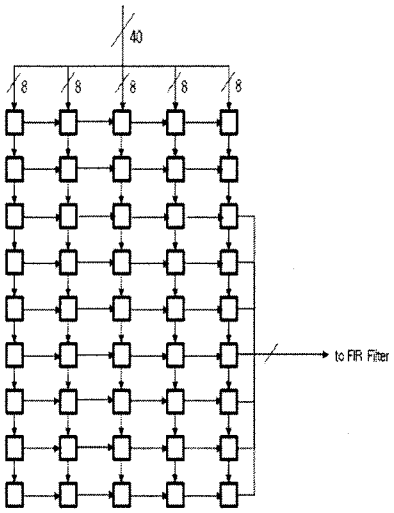

그림 8. 5x9 전치 배열의 구조

전치 배열 모듈을 통과하면 그림 1의 h, m, ee, ff 화소를 구하기 위해 6-tap 필터로 입력되어 결과가 레지스터에 저장되고 곧바로 4x9전치 배열에 저장되어 있던 화소들이 j 화소를 구하기 위해 6-tap 필터에 입력되어 진다.

마지막으로 평균 필터인 2-tap 필터를 사용하여 1/4화소를 생성하는데 순차적으로 입력받아 6-tap 필터가 수행되는 동시에 2-tap 필터가 수행된다.

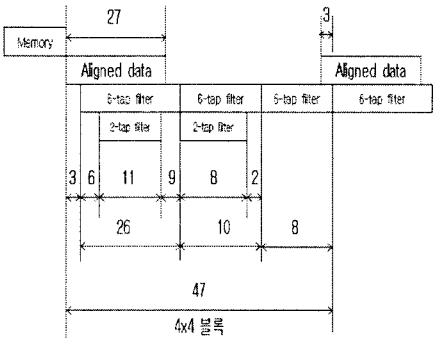

그림 9. 움직임 보상기의 타이밍도

그림 9는 제안한 움직임 보상기의 타이밍을 보여주고 있다. 메모리로부터 읽은 데이터는 정렬기

를 통해 3클럭 마다 데이터가 출력되어 5x9 전치 배열에 저장되며 동시에 2개의 6-tap 필터로 입력되어 26사이클 동안 연산된다.

6-tap 연산 결과를 4x9 전치 배열에 저장되며 정렬기 출력값이 6-tap 필터의 연산이 끝나면 5x9 전치 배열의 값이 다시 6-tap 필터로 입력되어 10사이클 동안 연산된다. 5x9 전치 배열의 연산이 끝나면 마지막으로 4x9 전치배열이 6-tap 필터로 입력되어 8사이클 동안 계산된다. 6-tap 필터는 하나의 매크로 블록이 처리될 때 까지 쉬지 않고 동작한다. 4x4 블록이 처리될 때까지는 입력 벡터값에 따라 30~47사이클이 소요된다.

IV. 성능 평가

움직임 보상의 핵심이 되는 6-tap 필터를 얼마나 사용하였는지와 최악의 경우 4x4 블록에 대한 수행 사이클을 비교하면 아래 표 1과 같다.

표 1. 회로 합성 결과

	[6]	[7]	본 논문
보간화소	16x16	16x16	16x16
수행사이클(4x4)	27	60	47
6-tap 필터	13	2	2

표 1에서 [6]의 구조는 4x4 블록을 수행하기 위해 27사이클이 걸리지만 6-tap 필터의 개수가 13개 사용되었고 [7]의 구조에서 필터의 개수는 동일하지만 수행 사이클에서 제안한 논문이 효율적이라고 할 수 있다.

V. 결 론

움직임 보상은 디코더 부분에서 시간적으로 압축된 영상을 복원하는데 사용되는 핵심블록이다. 본 논문은 H.264/AVC 디코더에 적용 가능한 움직임 보상에 효율적인 구조를 제안한다. 2개의 6-tap 필터는 데이터 입력 될 때 2클럭당 1번을 쉬며 데이터의 입력이 끝난 후 4x9전치 배열과 5x9전치 배열을 사용으로 쉬지 않고 동작한다. 논문에서 제안한 움직임 보상기는 HDL로 구현하고 ModelSim6.0a를 통해 검증하였다. 하나의 매크로블록을 처리하는데 최대 752 사이클이 소요되며 이는 H.264/AVC 디코더에서 효과적으로 처리 할 수 있을 것으로 사료된다.

참 고 문 헌

[1] Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification, May 2003, Joint Video Team.

[2] lain E.G. Richardson, "H.264 and MPEG-4",WILEY, 2003

[3] J. Ostermann, J.Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, T. Wedi, "Video coding with H.264/AVC:tools, preformance, and complexity", IEEE Circuits and System Magazine, Volume : 4, Issue : 1,First Quarter,2004,Pages:7-28

[4] H. Tseng, C. Chang and Y. Lin, "A Motion Compensator with Parallel Memory for H.264 Advance Video Coding," IEEE ISCAS Circuit and System, vol. 5, pp. 4558-4561, May 2005.

[5] 박기현, "코덱의 세계로의 초대," 홍릉과학출판사,2006

[6] H. Tseng, C. Chang and Y. Lin, "A Hardware Accelerator for H.264/AVC Motion Compensation," IEEE Signal Processing System Design and Implementation pp.214-219, Nov. 2005

[7] S. Wang, T. Lin, T. Liu and C. Lee, "A Motion Compensator with Parallel Memory for H.264 Advance Video Coding," IEEE ISCAS Circuit and System, vol. 5, pp. 4588-4561, May 2005.