메모리 접근 감소를 위한 움직임 예측기

*최민석 **이성원

광운대학교 컴퓨터 공학과

*hellohibye@kw.ac.kr **swlee@kw.ac.kr

Motion Estimation Architecture for Low Memory Access in H.264/AVC

*Choi Minseok **Lee Seong-Won

Dept. of Computer Engineering

Kwangwoon University

요약

움직임 추정(ME)은 동영상 압축에서 영상 화질과 인코더 속도에 대하여 중요한 역할을 하지만, 많은 수의 메모리 접근과 연산량이 발생한다. 기존의 움직임 추정 방법은 현재 프레임의 블록을 참조 프레임의 검색범위 내의 블록과 매칭하여 움직임 차이를 계산하여 움직인 위치를 추정하게 된다. H.264와 같은 최근의 압축 표준에서는 1/4화소 단위까지 움직임 예측을함으로써 영상 데이터 압축의 효율을 높일 수 있으나, 많은 양의 메모리 접근과 연산의 복잡도가 크게 증가하게 된다.

본 논문에서는 메모리 접근 횟수를 감소시키기 위하여 SAR(Search Area Reuse)알고리즘을 사용하여, 참조 프레임의 블록을 현재 프레임의 블록과 매칭하여 움직임 예측하는 방법을 제안한다. 본 논문에서 제안하고 있는 아키텍쳐는 현재프레임의 검색범위 내에 있는 데이타를 재사용함으로서 메모리 액세스를 줄일 수 있으며, 참조프레임의 한 블록당 1/4화소 단위까지의 연산을 한 번만 하게 되므로 메모리 접근 횟수 감소와 함께 연산의 복잡도도 줄일 수 있다.

1. 서론

H.264와 같은 동영상 압축의 표준은 동영상의 중복되는 데이터를 제거하기 위하여 화면 내 예측과 화면 간 예측을 한다. 인트라 예측은 공간적 중복성을 제거하여 압축 효율을 높이는 것이고, 압축율을 보다 효율적으로 높일 수 있는 인터 예측은 현재 프레임과 이전 프레임간의 중복성을 이용하여 블록매칭 방법을 사용한 움직임 추정을 한다.

쿼터픽셀 움직임 추정(ME)은 현재 프레임과 이전 프레임의 움직임 차이를 정화소단위부터 1/2화소 단위, 1/4화소 단위까지 함으로써 영상 데이터 압축의 효율을 높인다. 그러나 1/4화소 단위까지 움직임 추정을 하기 위해서는 많은 양의 메모리 접근과 복잡한 연산이 필요하게 된다.

본 논문에서는 메모리 접근 횟수 증가를 해결하기 위한 방법으로 검색 영역 재사용(SAR) 알고리즘을 사용하여 메모리 접근 횟수를 감소시킨다.

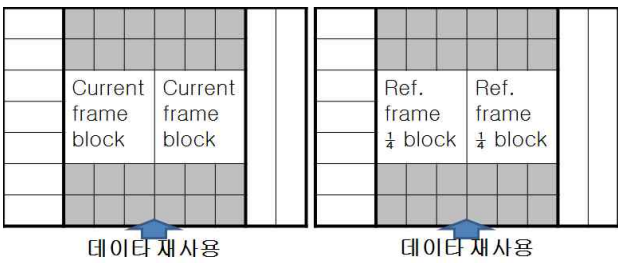

그림.1 (a) 기존의 SAR (b)논문에서 제안하는 SAR

기존의 SAR은 그림. 1(a)처럼 현재 프레임의 매크로블록(MB)의 연속적인 참조프레임의 검색영역내의 중복된 데이터를 재사용하는 방법[1]이다. 이 SAR알고리즘은 검색 영역내의 중복된 데이터를 재사용함으로써 메모리 접근 횟수를 크게 감소시킬 수 있다. 또한, 기존의 움직임 추정 방법은 현재 프레임의 MB에서 참조프레임의 검색 영역 내의 데이터들과 비교를 함으로써 움직임 벡터(MV)를 구했으나, 이러한 방법은 반복적으로 중복된 1/4화소 추정을 하기 위해 많은 양의 메모리 접근과 연산을 해야 한다.

본 논문에서는 1/4화소 추정을 위한 연산량과 메모리 접근 횟수를 감소시키기 위하여, 기존의 방법과는 반대로, 그림. 1(b)처럼 참조프레임의 4x4 블록을 1/4화소까지 추정한 후, 현재 프레임의 검색 영역내의 데이터들과 비교를 함으로써 MV를 구하는 방법을 제안한다.

본 논문은 4개의 절로 구성되어 있다. 2절에서는 본 논문에서 제안하는 움직임 추정 아키텍쳐를 설명하고, 3절에서는 제안한 아키텍쳐의 성능분석을 기술하며, 4절에서는 결론을 제시한다.

2. 제안하는 움직임 추정 Architecture

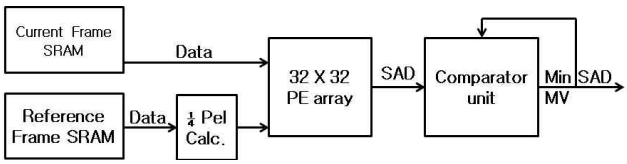

그림. 2 제안하는 아키텍쳐의 블록 다이어그램

그림. 2는 SRAM과 1/4 pel calculator unit, 32x32 PE array,

comparator unit으로 구성되어 있는 본 논문에서 제안하는 아키텍처의 블록 다이어그램이다.

Current Frame SRAM에는 현재 프레임의 화소 데이터가 저장되어 있고, Reference Frame SRAM에는 참조프레임의 화소 데이터가 저장되어 있다. 1/4 Pel Calc. unit은 참조프레임의 화소 데이터를 받아서 1/4화소 단위까지 추정을 한 후 정화소, 1/2화소, 1/4화소의 데이터를 계산하여 32x32 PE array에 보낸다. 32x32 PE array는 현재 프레임의 4x4블록 64개와 1/4화소 단위까지 추정을 한 참조 프레임의 4x4블록 하나와 SAD계산을 한다. 1/4화소는 하나의 화소당 16개이므로 위치별로 한 번씩 16번의 SAD계산을 한다. 64개의 현재 프레임 4x4블록은 comparator unit에서 최소의 SAD값을 계산하여 MV를 구한다.

가. 데이터 재사용을 위한 데이터 스케쥴링 방법

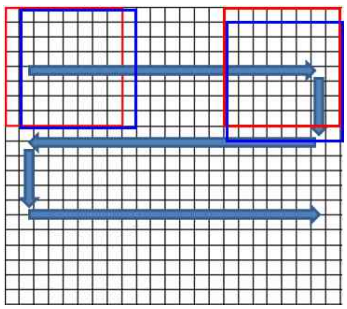

그림. 3 현재 프레임 스캔

검색 범위가 [-16~+15]인 경우, 참조 프레임 하나의 4x4블록과 비교를 하는 현재 프레임의 4x4블록은 64블록이다. 참조 프레임 하나의 4x4블록을 좌에서 우로, 그리고 우에서 좌로 이동을 하면서 현재 프레임과 비교를 할 때, 한 번의 이동시마다 기존의 56블록의 데이터는 그대로 사용하고, 새로운 8블록의 화소 데이터를 메모리로부터 읽어온다.

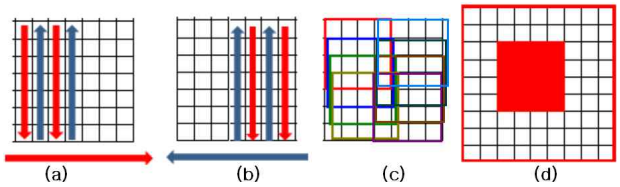

그림. 4 참조 프레임 스캔

64블록의 현재 프레임에 대응되는 참조프레임의 화소는 그림.4(c)처럼 행(row)과 열(column)이 다른 4x4블록 16개로 전체 7x7블록의 크기를 가진다. 참조프레임은 1/4화소 단위로 추정을 해야 하기 때문에 4x4블록 16화소의 움직임을 1/4화소 단위로 동시에 추정하기 위해서는 그림 4.(d)처럼 9x9화소가 필요하다. 또한, 계산 시간을 줄이기 위하여 7x7블록을 동시에 1/4화소 단위로 추정을 하면, 12x12화소가 필요하다. 4x4블록단위로 메모리에서 읽어오는 방식이므로 16블록을 메모리에서 읽어온다. 한 번 이동시마다 12블록은 그대로 사용하고, 새로운 4블록의 화소 데이터를 메모리로부터 읽어온다.

나. 프레임 메모리 구성

SRAM으로부터 화소 데이터를 읽어 올 때, 한번에 4x4블록 단위인 128비트 단위로 읽어올 수 있도록 프레임 메모리의 데이터 배열을 구성한다.

두 개의 SRAM 모듈은 현재 프레임과 참조 프레임의 화소 데이터를 저장하는데 사용이 된다. 각 SRAM 모듈은 하나의 4x4블럭이 한 클럭에 출력할 수 있도록 128bit word SRAM으로 구성한다.

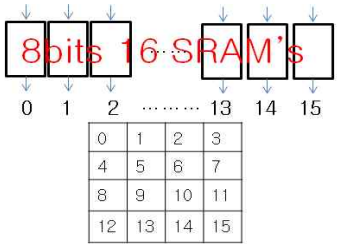

그림. 5 Current Frame과 Ref. Frame SRAM

다. 프로세스 유닛

그림. 6는 process unit이다.

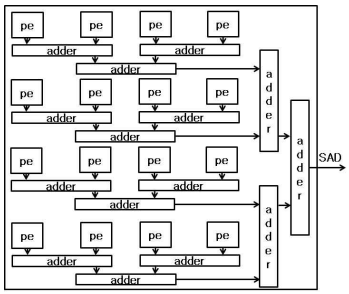

그림. 6 process unit

process unit은 4x4 PE와 15개의 addr로 구성되어 있다. 이 process unit 64개가 모여 한 번에 64블록의 SAD를 계산하게 된다. 참조프레임의 4x4블록이 현재프레임 4x4블록의 검색 영역 밖으로 이동하기 전까지는 현재프레임의 32x32화소 데이터는 process unit에 고정되어 있다. 참조프레임의 4x4블록이 현재프레임 4x4블록의 검색영역 밖으로 이동하게 되면 그림. x와 같은 방법으로 현재프레임의 주변 4x4블록 8개를 읽어와서 process unit 8개를 채운다.

라. comparator unit

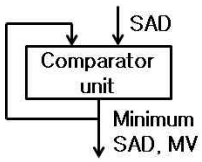

그림. 7 comparator unit

그림. 7은 SAD값을 비교하여 최소SAD값을 가지는 MV를 저장하도록 하는 comparator unit이다. 참조프레임의 4x4블록이 현재프레임 4x4블록의 검색영역 밖으로 나가기 전까지 SAD값을 비교하여 MV를 저장하게 된다. 이 유닛도 프로세스 유닛과 마찬가지로 64개가 모여 한 번에 64개 블록의 SAD값을 비교하게 된다.

3. 성능분석

본 논문에서 제안하고 있는 아키텍처의 성능을 분석해보면, 참조프레임의 첫 번째 7x7블록을 1/4화소단위까지 추정하기 위해서는 4x4블록 16개가 필요하고 16개의 4x4블록을 메모리로부터 읽어오는 데 16클록이 소요된다. 7x7블록을 1/4화소 단위로 보간하기 위해서는 처음 한 줄에 9클록과 그 다음 줄마다 3클럭씩 총 27클록이 소요된다. 그 다음 7x7블록을 1/4화소 단위로 보간하기 위해서는 주변블록 4개를

메모리로부터 읽어오기 위해 4클럭이 소요된다. 그러므로 처음 한번만 43클럭이 소요되고 그 다음부터는 31클럭이 소요된다. 참조 프레임의 블록이 이동한 경우, 현재 프레임 메모리에서 4x4블록 8개의 데이터를 읽어오고 8클럭이 소요된다. 이것은 병렬로 처리하게 되므로 무시할 수 있는 클럭이다. 32x32 PE array에서는 SAD를 계산하는데 16클럭이 소요된다. comparator unit 또한 16클럭이 소요된다. QCIF의 경우, 176x144해상도로 4x4블록 1584개로 구성되어 있으며, 한 프레임당 12+31*44*36+32 = 49148클럭이 소요된다.

4. 결론

본 논문에서는 H.264/AVC를 위한 움직임 추정기의 구조를 기술하고, 하드웨어 설계를 위한 구조를 제안하였다. 기본적으로 SAR알고리즘을 사용하여 메모리 접근 횟수를 감소시켰으며, 참조 프레임 블록의 1/4화소 단위로 추정을 한 후 현재 프레임의 블록과 매칭하는 방법으로 메모리 접근 횟수와 함께 연산량의 감소도 꾀하였다.

Acknowledge

본 논문은 서울시 산학연 협력사업(10560,10570)및 교육과학기술부의 재원으로 한국연구재단의 지원(2009-0088064)에 의해 연구 되었음

References

1. Heejun Shim, Chong-Min Kyung, "Selective Search Area Reuse Algorithm for Low External Memory Access Motion Estimation", IEEE Trans. Circuits Syst. Video Technol., vol. 19, no. 4, pp. 1044-1050.(2009)
2. Seung-Man Pyen, Keyong-Yuk Min, Jong-쫌 Chong, Alberto L. Sangiovanni-Vincentelli, "An Efficient Hardware Architecture for Full-Search Variable Block Size Motion Estimation in H.264/AVC"
3. 임영훈, 이대준, 정용진, "MPEG-4 AVC를 위한 고속 인터 예측기의 하드웨어 구현", 한국통신학회논문지 vol. 30 no. 3c, pp. 102-111.(2005)