H.264/AVC 디코더의 움직임 보상을 위한 메모리 접근 감소 기법

정회원 박 경 오*, 종신회원 홍 유 표**°

Memory Access Reduction Scheme for H.264/AVC Decoder Motion Compensation

Kyoungoh Park *Regular Member*, Youpyo Hong *Lifelong Member*

요 약

H.264/AVC 디코더의 하드웨어 구현 시 실시간 동작을 위한 가장 큰 장애 요소 중 하나인 외부 메모리 엑세스량을 크게 줄인 움직임 보상 기법을 제안한다. H.264/AVC 디코더의 움직임 보상용 참조 영상은 큰 용량 때문에 대게 외부 메모리에 보관되며, 참조 영역은 수시로 디코더 코어 내부로 읽혀지게 되는데, 단순히 참조 영역 단위별 순차적 메모리 접근을 할 경우 그 데이터 엑세스 량은 디코더의 실시간 동작이 불가능할 정도로 막대할 수가 있다. 본 논문에서는 참조 영역을 매크로블럭 단위로 분석하여 가급적 적은 메모리 엑세스로 필요한 참조 영역을 읽어 들이는 방식을 제안하고 있으며, 실험 결과 제안된 움직임 보상 기법은 단순한 순차적 참조 블록별 데이터 접근 방식 대비 외부 메모리 사용 대역폭을 약 30% 감소시킴을 확인할 수 있었다.

Key Words : H.264/AVC, Motion Compensation, Decoder, SDRAM, Codec

ABSTRACT

In this paper, a new motion compensation scheme to reduce external memory access frequency which is one of the major bottlenecks for real-time decoding is proposed. Most H.264/AVC decoders store reference pictures in external memories due to the large size and reference blocks are read into the decoder core as needed during decoding. If the reference data access is done for each reference block in decoding sequence, the memory bandwidth can be unacceptable for real-time decoding. This paper presents a memory access scheme for motion compensation to read as many reference data as possible with reduced memory access frequency by analyzing reference data access pattern for each macroblock. Experimental results show that the proposed motion compensation scheme leads to approximately 30% improvement in memory bandwidth requirement.

Ⅰ. 서 론

최근 방송, 통신, 가전 등 대부분의 차세대 멀티미디어 동영상 표준으로 채택된 H.264/AVC는 압축률 측면에서는 현저하게 향상이 되었으나 연산량이나 복잡도 또한 크게 증가하여 구현 측면에서는 많은 난제를 제시하고 있다. 특히, 압축률 면에서 큰 영향을 미치는 움직임 예측에 있어 다양한 가변 블록 크기 지원과 사분 화소 지원이 됨에 따라, 움직임 보상 연산은 기존 압축 표준 대비 월등히 많은 량의 데이터를 필요로 하게 되었으며, 이는 곧 데이터를 저장하는 외부 메모리의 대역폭 수요가 크게

* 동국대학교 전자공학과
** 동국대학교 IT학부 전자공학전공 (° : 교신저자, yhong@dgu.edu)
논문번호 : KICS2009-03-089, 접수일자 : 2009년 3월 4일, 최종논문접수일자 : 2009년 3월 17일

증가하였음을 의미한다.

이러한 외부 메모리 엑세스량을 줄이기 위한 연구는 이미 MPEG2 압축 표준의 구현 때부터 시작되었다. Li[1]는 MPEG2 디코더를 위한 DRAM대역폭의 효과적 활용을 위한 버스 아비터를 제안하였고 Ling[2]은 MPEG2디코더에 대하여 미리 지정된 메모리 접근 순서에 의해 최대 메모리 대역폭을 줄이기 위한 메모리 맵을 제안하였다. Kim[3] 은 선형적 주소체계에 따른 열-활성화에 기인한 오버헤드 사이클 수를 줄이기 위하여 선형적 주소를 물리적 주소로 바꾸는 아키텍처를 제안하였다. Park[4]는 HDTV디코딩 응용을 위해 히스토리 기반 메모리 컨트롤러, 즉 매 움직임 보정 연산마다 후속 동작의 예측을 통해 메모리 엑세스 지연시간을 줄이는 방법을 제안하였다.

H.264의 움직임 추정을 위한 메모리 엑세스 최적화 관련 연구는 비교적 최근 시작되었으며 Wang[5]과 Tsai[6]은 다른 서브 블록간 데이터를 최대한 공유하고 인터폴레이션을 위해 꼭 필요한 데이터만을 읽어 들이는 방식을 제안하였다.

본 논문에서는 범용 SDRAM 컨트롤러를 그대로 사용하면서 매크로 블록 단위로 움직임 보상 연산에 필요한 참조 영역들을 최대한 동시에 많이 읽어 들임으로서 SDRAM엑세스 빈도를 크게 줄이는 방법을 제안하였다.

Ⅱ. SDRAM 기초

그림 1은 비디오 코덱의 프레임 버퍼 용도로 많이 사용되고 있는 SDRAM의 기본 구조를 보여 주고 있다. SDRAM은 보통 다수의 뱅크를 가지며, 각 뱅크는 고유의 행-디코더를 가진다. SDRAM의 고 용량화로 인하여 행-디코더와 열-디코더의 디코딩 지연시간이 늘어나게 되었고 이에 따른 SDRAM 접근 시간의 증가로, 주소 디코딩 로직은 파이프라인화 되었고, 동일 열 내의 연속된 데이터 접근시는 버스트 모드를 이용하게 된다.

그림 2는 버스트 길이 4의 SDRAM 쓰기 사이클의 타이밍도를 보여주고 있다. CS(Chip Select)와 RAS(Row active strobe)가 '로우'가 되면 열-활성화 상태가 되며 이 때 입력되는 주소값이 열-주소가 된다. 열-디코더가 열-주소를 디코딩 하는 시간을 통상 tRCD(RAS to CAS delay)라고 하며, 이 tRCD 이후 CAS(Column active strobe)와 WE가 '로우'로 떨어지게 되면 쓰기 모드로 진입하며, 이

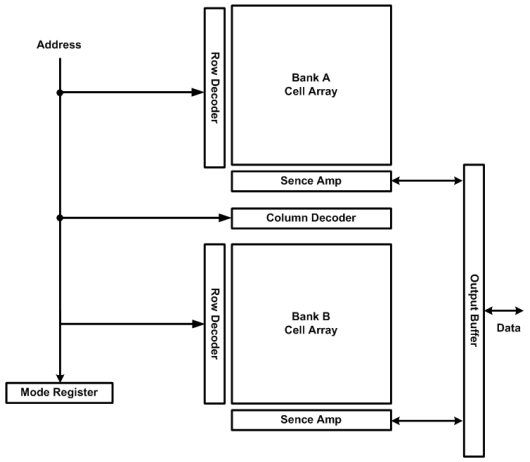

그림 1. SDRAM 구조

때 입력되는 주소 값이 열-주소가 되고, 데이타도 동시에 입력된다. 버스트 모드이기 때문에 정해진 길이만큼 데이터가 입력되면 쓰기 사이클이 종료된다. 버스트 길이 등의 설정은 버스트 동작 직전 모드 레지스터에 의해 설정된다.

SDRAM 읽기 동작시는 메모리 셀 내부의 데이터 페치 시간 때문에 쓰기와는 다소 다른 타이밍도를 보여 주고 있다. 그림 3.을 보면 열-활성화 상태로의 진입은 동일하지만, tRCD 후 열-활성화 입력시 CAS는 '로우', WE는 '하이'인 것을 볼 수 있다. 또한, 메모리 셀에서 데이터가 나오기까지의 시간을 Cas Latency라고 하며, 그림 3. 에서는 Cas Latency = 3 인 경우를 나타내고 있다. Cas Latency는 SDRAM의 동작 주파수 및 제조 공정에 따른 편차는 존재하지만, 2 또는 3인 경우가 대부분이다.

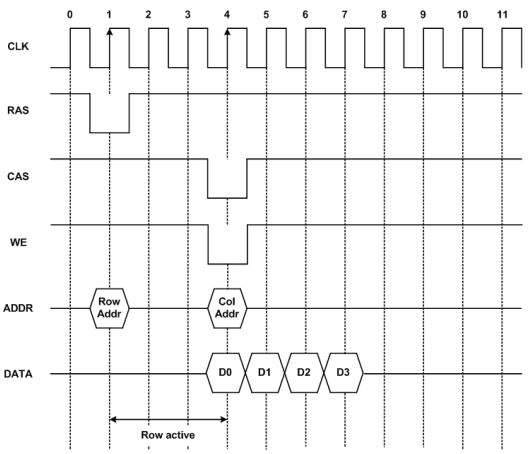

그림 2. SDRAM 쓰기 사이클 타이밍도

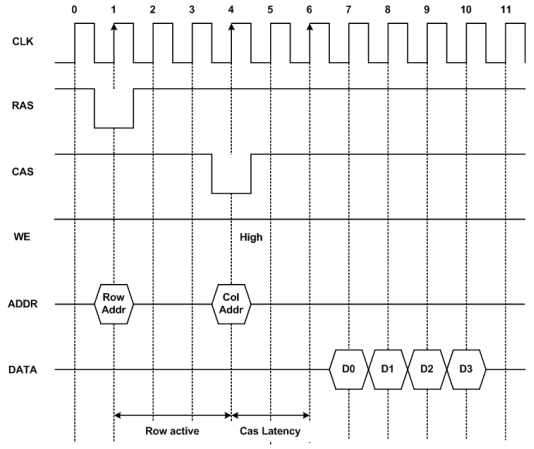

그림 3. SDRAM 읽기 사이클 타이밍도

그림 2와 그림 3은 일반적인 SDRAM 사이클을 나타내고 있으나, 본 논문에서는 AHB를 사용한 SDRAM 콘트롤러(이하 SDC)를 사용하고 있기 때문에, AHB 마스터가 버스의 사용권한을 얻고, 트랜잭션을 개시한 시점부터 행-활성화까지의 추가 오버헤드가 SDRAM 접근 사이클에 추가된다. 따라서 버스트 4 읽기 시는 (SDC의 소요 클럭 + tRCD + CAS Latency + 버스트 길이) 만큼의 클럭이 필요하게 되며 본 논문에서 사용한 SDC의 소요클럭은 5클럭, tRCD는 4클럭, CAS Latency = 3이기 때문에, 합산 시 총 16클럭이 소요되게 된다. 버스트 8 읽기 시에는 위의 계산법을 사용하면 총 20클럭이 소요되게 된다.

만약 버스트 모드를 사용하지 않게 된다면 (SDC의 소요클럭 + tRCD + CAS Latency + 1)*(Data의 개수) 만큼의 클럭이 필요하게 되므로 상당한 오버헤드가 발생하게 된다.

따라서 프레임 버퍼에 적용되는 메모리맵은 버스트 엑세스를 최대한 활용하여 동일 데이터의 읽기 및 쓰기 동작을 위해 소요되는 클럭 사이클의 수효를 최소화 하여야 한다.

Ⅲ. H.264/AVC 움직임 추정

H.264/AVC는 기존의 MPEG2/MPEG4 대비 여러 측면에서 압축률 개선 기능을 추가 했는데, 인터 예측에 있어서는 16*16, 16*8, 8*16, ... 4*8, 4*4의 다양한 파티션 예측 모드, 그리고 보다 정확한 정밀도의 1/2화소 및 1/4화소 단위 움직임 예측이 대표적인 변화이다. 이렇게 세분화되고 정밀도가 향상 된 움직임 예측은 압축률 면에서는 월등한 성능을 제공하지만, 구현 측면에서는 다양한 조합의 계산은 물론 그러한 계산에 소요되는 많은 수효의 픽셀 데이터를 수급해야 하는 구현상의 부담이 추가되었다. 가령, 4*4 파티션으로 구성된 1/2화소 예측의 경우 움직임 예측을 위해 필요로 하는 데이터의 량은 (2+4+3) * (2+4+3) = 81 픽셀이며, 여기서 2와 3은 각기 좌/우 또는 상/하 주변 픽셀로서 반화소 픽셀 계산을 위해 해당 4*4 픽셀에 추가하여 필터에 적용하기 위하 픽셀이다. MPEG4 ASP에서도 유사한 필터가 사용되기는 했으나 내부 픽셀을 블록 경계를 중심으로 대칭적으로 복제 사용함으로서 부수적으로 메모리 엑세스를 엑세스하지는 않았다. 그러나, H264/AVC에서는 블록 외곽 픽셀을 실제로 필터에 적용하여 보다 정확한 1/2, 1/4 화소 계산을 하도록 고안되었으며, 이는 막대한 외부 메모리 엑세스 량을 의미하므로 H264/AVC디코더의 실시간 동작에 있어 큰 장애요소로 등장하였다.

Ⅳ. 저 메모리 대역폭 움직임 추정 회로

본 논문에서 제안한 움직임 추정 방식의 기본 아이디어는 그게 다음 두 가지 사항이 기초하고 있다.

- 단일 매크로블럭에 속하는 움직임 벡터간의 좌표 차이는 통상적으로 수 픽셀 이내이다.
- 다수의 소폭 버스트보다 소수의 큰 버스트 엑세스가 크게 효율적이다.

단일 매크로블럭에 대한 움직임 추정 시 엑세스 대상 데이터를 포함하는 참조 블록의 위치는 비교적 근접해 있는 경우가 많아서, 각 참조 블록을 분리 된 버스트 엑세스로 엑세스 하지 않고, 다소 증가된 크기의 단일 버스트 엑세스로 엑세스 한다면 전체적 엑세스 사이클 수에서 유리할 수 있다는 것이 첫 번째 아이디어였으며, 실제로 대부분 SDRAM 컨트롤러의 경우 이러한 잇점이 발휘됨을 확인할 수 있었다. 가령 버스트 4는 18클럭, 버스트 8은 22클럭이 소요되는 AHB 호환 SDRAM 컨트롤의 경우 큰 버스트 엑세스를 이용하여 처리할 수 있는 범의의 데이터는 최대한 큰 규모의 버스트 엑세스를 이용하는 것이 전체 사이클 수에서 유리한 것으로 관찰 되었다.

본 논문에서는 소위 윈도우 개념을 도입하여, 단

일 매크로블럭에 속하는 움직임 벡터에 대하여 가급적 많은 수효의 데이터를 포함하여 읽어 들일 수 있는 버스트 엑세스 경우를 찾아 내어 적용을 하고, 그 윈도우에 의하여 처리 되지 못한 움직임 벡터를 처리해 나가는 방식으로 참조 영상을 읽어 들인다.

이러한 기본적 아이이어의 구현에 있어서 다음가 같은 실제적 문제들이 해결되어야 했다.

- 다양한 모드 조합에 있어 어떠한 모드에 대하여 윈도우 방식을 적용할 것인가?
- 윈도우의 크기와 개수는 어떻게 결정되는가?

H.264 인터 예측의 경우 가능한 파티션 16*16, 16*8은 한 개 행에 대해서 2+16+3의 총 21픽셀 데이터를 필요로 하므로 최소 버스트 8엑세스를 필요로 하며, 이 21개 픽셀의 시작 주소가 4의 배수가 아닌 경우에도 버스트 8로 모든 픽셀의 엑세스가 충분하다. 따라서 이 두 파티션의 경우는 윈도우 적용이 필요 없다.

8*8 파티션에 대한 윈도우 적용 알고리즘의 첫 단계는 8*8 파티션 내 서브 파티션의 종류에 상관없이 모든 움직임 벡터를 4*4 단위로 표시하는데, 이는 후속 처리 과정을 4*4에 맞추어 단순화 시키기 위함이다.

```
insert mvs for all 4*4 sub-partitions into mv_queue;

while(mv_queue not empty){
  col_adrs = min{mvxs in mv_queue};

  for (i=0 to 15) {
    if (col_adrs <= mvx[i] && mvx[i]+8 <= col_adrs) {
      mv_in_window[i] = 1; remove mv[i] from mv_queue;
      for (j=0 to 8) insert mvy[i]+j into row_adrs_queue;
    }
  }


  for each row_adrs in row_adrs_queue {
    reg[31:0]

 = burst8 data from sdram[row_adrs][col_adrs]
    for (i=0 to 8) {
      if (mvy[0]<=row_adrs <= mvy[0]+8 & mv_in_window[0]=1)
        sram0[row_adrs-mvy[0]][i]=reg[(mvx[0]-col_adrs)+i];
      if (mvy[1]<=row_adrs <= mvy[1]+8 & mv_in_window[1]=1)
        sram1[row_adrs-mvy[1]][i]=reg[(mvx[1]-col_adrs)+i];
        ...
      if (mvy[15]<=row_adrs<=mvy[15]+8 &
mv_in_window[15]=1)
        sram15[row_adrs-mvy[15]][i]=reg[(mvx[15]-col_adrs)+i];
    }
  }
}
```

그림 4. MC용 윈도우 설정 및 데이터 분배 알고리즘

다음으로 윈도우의 범위를 결정하는 단계에서는 가로 방향에서 시작 지점은 처리해야 할 움직임 벡터 중 가장 좌측 움직임 벡터를 택하였다. 세로 방향에 있어서는, 선택된 x-범위 내에 속하는 움직임 벡터 서브블럭이 포함되는 라인들을 포함시켰다.

이렇게 윈도우의 범위가 결정되고 나면, 윈도우에 포함되는 각 행에 대하여 SDRAM으로부터 버스트 4에 해당하는 데이터를 읽어 들인다. 그리고 16개 서브 블록별로 이렇게 읽어 들인 데이터를 분배하는데, 이 때 분배 해당 여부는 각 서브블럭의 움직임 벡터 좌표와 읽어 들인 데이터의 좌표를 대조함으로서 판별된다. 그림 4는 전술된 알고리즘을 수도코드의 형태로 보여주고 있다.

V. 실험 결과

제안된 알고리즘은 Verilog를 이용하여 구현되었으며, 네 개 영상에 대하여 다양한 QP별로 움직임 보상회로의 외부 메모리 대역폭 성능 향상 효과를 측정하였다.

우선 표 1은 각 영상들에 대해 세 가지 QP별로 인코딩 하였을 때 인터 모드로 예측된 경우 파티션의 분포를 보여주고 있다. 인코딩 조건은 GOP 10, 탐색 영영 ±64 화소 이며, QP는 10, 25, 40 세 경우에 대하여 테스트 하였다. 테스트 결과, QP가 낮을수록 서브파티션이 적용될 확률이 높게 나타났으며, 이는 영상의 정밀도가 높을수록 세밀한 단위의 움직임 예측이 보다 큰 효용을 발휘했기 때문으로 판단된다.

표 2는 제안 알고리즘 적용 시 각각의 인코딩 스

표 1. QP별 파티션 분포 (JM12.2)

Sequence name	QP	Partition(%)						
		16x16	16x8	8x16	8x8	8x4	4x8	4x4
Flower	10	44.49	9.97	6.93	19.05	8.11	7.48	3.47
	25	57.29	10.06	5.79	14.25	5.56	4.93	2.09
	40	62.57	9.45	5.85	10.92	5.32	3.99	1.87
Foreman	10	23.57	10.07	10.75	25.02	13.70	11.32	5.53
	25	39.00	12.32	12.17	18.36	8.33	6.77	3.02
	40	68.85	11.31	11.72	5.11	1.65	1.08	0.26
Mobile	10	37.23	10.19	8.72	22.05	9.55	9.34	2.91
	25	39.68	10.82	9.10	20.58	8.23	8.71	2.85
	40	46.92	11.72	10.71	16.60	5.53	6.93	1.58
Paris	10	43.31	5.47	5.80	21.00	9.53	9.45	5.41
	25	66.63	5.53	5.86	11.52	4.25	4.84	2.35
	40	74.62	5.14	5.03	8.30	2.64	2.95	1.31

표 2. QP 별 윈도우 적용 개수

Sequence name	QP	window(%)				MAX	AVG
		1	2	3	4		
Flower	10	88.02	11.24	0.74	0.00	3	1.12
	25	94.61	5.14	0.25	0.00	3	1.05
	40	97.69	2.31	0.00	0.00	2	1.02
Foreman	10	95.76	4.18	0.06	0.00	3	1.04
	25	97.85	2.00	0.15	0.00	3	1.02
	40	93.10	6.90	.000	0.00	2	1.06
Mobile	10	89.21	9.08	1.52	0.19	4	1.12
	25	90.19	8.54	1.27	0.00	3	1.11
	40	92.26	7.74	0.00	0.00	2	1.07
Paris	10	98.40	1.60	0.00	0.00	2	1.01
	25	97.82	2.18	0.00	0.00	2	1.02
	40	95.32	4.68	0.00	0.00	2	1.04

트림에 대하여 움직임 보상 연산을 위해 몇 번의 윈도우가 적용되었는지에 대한 측정 결과를 보여주고 있다. 90%이상의 경우 한 번의 윈도우 적용으로 움직임 보상이 필요로 하는 모든 참조 영역 엑세스가 완료되었음을 관찰할 수 있다.

표 3은 제안 알고리즘 적용시 한 개 매크로 블록에 대해 소요된 사이클 수에 대한 측정 결과를 보여주고 있다. 약 50%이상 매크로블럭에 대해 16*16 파티션의 움직임 예측이 되었던 Forman영상 QP 40의 경우를 제외하고는 대략 30%가량의 사이클 수 절감 효과가 있었음을 관찰할 수 있다.

표 3. 매크로블럭 당 MC의 메모리 이용 사이클 수 비교

Sequence name	QP	Bit rate (Mbps)	Original data cycles	Data cycles use window	Saved (%)
Flower	10	**9.29**	755	478	36.7
	25	**4.63**	657	467	28.9
	40	**1.58**	626	465	25.7
Foreman	10	**7.62**	913	491	46.2
	25	**1.89**	751	488	35.0
	40	**0.40**	525	478	8.9
Mobile	10	**12.80**	791	495	37.4
	25	**6.13**	771	490	36.4
	40	**1.85**	694	485	30.1
Paris	10	**7.12**	780	462	40.8
	25	**2.26**	639	462	27.7
	40	**0.69**	568	457	19.5

Ⅵ. 결 론

본 논문에서는 H.264/AVC 스트림의 실시간 디코딩에 있어 가장 큰 장애 요소 중 하나인 움직임 보상 회로의 외부 메모리 엑세스 빈도를 줄이는 기법이 제안되었다. 단일 메크로 블록에 속하는 움직임 벡터를 분석하여 가장 많은 데이터를 적은 수효의 버스트 엑세스를 통하여 처리하는 방식으로 평균 30%정도의 메모리 대역폭 감소 효과를 이루었음을 실험을 통하여 증명하였다.

참 고 문 헌

〔1〕 J. Li and N. Ling, "Architecture and Bus-Arbitration Schemes for MPEG-2 Video Decoder," *IEEE Trans. Circuits and Systems for Video Technology,* vol. 9, no. 5, pp. 727-736, Aug. 1999.

〔2〕 N. Ling, N. Wang, D. Ho, "An Efficient Controller Scheme for MPEG-2 Decoder", *IEEE Trans. On Consumer Electronics,* vol. 44, no. 2, pp. 451-458, May. 1998.

〔3〕 H. Kim and I. Park, "High-Performance and Low-Power Memory-Interface Architecture for Video Processing Applications,"*IEEE Trans. Circuits and Systems for Video Technology,* vol. 11, no. 11, pp. 1160-1170, Nov. 2001.

〔4〕 S. Park, Y. Yi and I. Park, "High Performance Memory Mode Control for HDTV Decoders", *IEEE Trans. On Consumer Electronics,* vol. 49, no. 4, pp. 451-458, Nov. 2003.

〔5〕 R. Wang, M. Li and Y. Zhang, "High Throughput and Low Memory Access Sub-pixel Interpolation Architecture for H.264/AVC HDTV Decoder", *IEEE Trans. On Consumer Electronics,* vol. 51, no. 3, pp. 1006-1013, Nov. 2005.

〔6〕 C. Tsai, T. Chen, T. Chen and L. Chen, "Bandwidth optimized motion compensation hardware design for H.264/AVC HDTV decoder," *Proc. Int. Symp. Circuits and Systems*, vol.2.pp273-276, Aug. 2005

박 경 오 (Kyoungoh Park) 정회원

2009년 2월 동국대학교 전자공학과 학사

2009년 3월~현재 동국대학교 전자공학과 석사 과정

<관심분야> 비디오 코덱 칩 설계

홍 유 표 (Youpyo Hong) 종신회원

1991년 2월 연세대학교 전기공학과 학사
1993년 5월 University of Southern California 전기공학과 석사
1998년 8월 University of Southern California 컴퓨터공학과 박사
1998년 7월~1999년 2월 Synopsys, Hillsboro, Senior Engineer
1999년 3월~현재 동국대학교 IT학부 전자공학전공 부교수
<관심분야> 멀티미디어 칩 설계, SOC 설계