H.264/AVC용 매크로블록 겹침 기법에 기반한 디블록킹 필터의 설계

김원삼* · 손승일*

Design of A Deblocking Filter Based on Macroblock Overlap Scheme for H.264/AVC

Won-sam Kim* · Seung-il Sonh*

이 논문은 2008년도 한신대학교 연구비 지원에 의해 수행되었음.

요 약

H.264/AVC는 블록킹 현상을 제거하기 위해 디블록킹 필터를 채용한 영상 이미지의 압축을 위한 새로운 국제 표준이다. 본 논문에서는 H.264/AVC에 존재하는 디블록킹 필터의 효율적인 아키텍처를 제안한다. 이웃한 4×4 블록 사이의 데이터 종속성을 이용하여 메모리의 사용량을 줄이고, 디블록킹 필터 처리의 쓰루풋을 향상시켰다. 본 논문에서 설계된 디블록킹 필터는 매크로블록 내에서는 수평 필터링과 수직 필터링을 파이프라인 방식으로 수행하고, 매크로블록 간에는 겹침 방식을 채용함으로써 병렬성을 한층 향상시켰다. 구현 결과는 기존의 디블록킹 필터와 비교할 때 1.95에서 4.73배까지 성능을 향상시키는 것으로 나타났다.

따라서 본 논문에서 제안한 디블록킹 필터의 아키텍처는 고해상도 비디오 응용에서 실시간으로 디블록킹을 수행할 수 있을 것으로 예견된다.

ABSTRACT

H.264/AVC is a new international standard for the compression of video images, in which a deblocking filter has been adopted to remove blocking artifacts. This paper proposes an efficient architecture of deblocking filter in H.264/AVC. By making good use of data dependence between neighboring 4x4 blocks, the memory size is reduced and the throughput of the deblocking filter processing is increased. The designed deblocking filter further enhances the parallelism by simultaneously executing horizontal and vertical filtering within a macroblock in pipeline method and adopting overlap between macroblocks. The implementation result shows that the proposed architecture enhances the performance of deblocking filter processing from 1.75 to 4.23 times than that of the conventional deblocking filter.

Hence the proposed architecture of deblocking filter is able to perform real-time deblocking in high-resolution(2048x1024) video applications.

키워드

H.264/AVC, Deblocking filter, Blocking artifact, Real-time, Data dependency, Pipeline, Parallelism

* 한신대학교 정보통신학과

접수일자 : 2007. 11. 21

Ⅰ. 서 론

H.264/AVC 비디오 압축은 비디오 압축을 위하여 가장 최근 개발된 국제표준이며, 전송률-왜곡 효율(Rate-distortion efficiency)에서 기존의 비디오 압축 알고리즘보다 우수한 성능을 보여 다양한 멀티미디어 응용 분야에 사용될 것으로 예견되는 기술이다[1].

MPEG-4 표준의 경우에는 DCT와 양자화 과정을 통하여 발생하는 블록화 현상을 효율적으로 처리하기 위해 후처리로서 부호화 과정 및 복호화 과정과 무관하게 복호화된 영상에 대해 선택적으로 필터링을 한다. 하지만, 이보다 두 배 이상의 압축 코딩 표준인 H.264/AVC는 주관적 화질의 개선과 부호화 효율의 개선을 위하여, 부호기 및 복호기 내부에 루프 디블록킹 필터(In-Loop Deblocking Filter)를 위치시킨다. H.264/AVC는 4×4블록을 기반으로 한 정수형 변환과 가변 블록 사이즈의 움직임 보상을 이용하기 때문에, 8×8 블록 단위의 DCT 변환을 기반으로 하는 기존의 비디오 압축 알고리즘과는 다른 블록화 현상이 나타난다. 이러한 블록화 현상을 제거하기 위한 디블록킹 필터는 H.264/AVC에서 높은 압축비와 개선된 화질을 얻기 위한 중요한 기술로 인식되고 있다[2].

블록화 현상을 제거하는 디블록킹 필터를 채용하여 압축된 영상은 필터 처리를 하지 않은 채 압축된 영상과 비교하여 5～10%의 비트율 절감 효과가 있다는 실험 결과를 발표하였다[3]. 그리고 디블록킹 필터는 H.264/AVC 복호기에서 복호 연산량의 1/3을 점유할 정도로 연산량이 많은 블록이다[4]. 이러한 근거를 바탕으로 판단해 볼 때, 영상 복호기의 실시간 처리를 위한 H.264/AVC 디블록킹 필터의 고속화 연구는 중요하게 인식되고 있다[6].

이에 본 논문에서는 향후 동영상과 관련된 멀티미디어에서 보다 적은 데이터를 더 빠르게 처리할 경우 만족스러운 화질을 얻기 위해 사용되는 디블록킹 필터를 적은 메모리를 사용하여 고속으로 처리할 수 있는 필터 처리 아키텍처를 제안하였다.

2장에서는 디블록킹 필터와 관련된 연구에 대해서 알아보고, 3장에서는 디블록킹 필터의 기본 개념에 대해 설명하며, 4장에서는 본 논문에서 제안하는 디블록킹 필터 설계에 대해 설명하고, 5장에서는 설계 결과의 비교 분석을 수행한다. 마지막으로 6장에서는 결론을 맺는다.

Ⅱ. 관련 연구

논문 [5]는 프로그램 가능한 1차원 필터의 구조를 사용하지만 수평 방향의 디블록킹 필터링 수행을 완료한 후에 수직 방향으로 디블록킹 필터링을 수행한다. 따라서 수평 필터링을 완료한 데이터를 이후 수직 필터링에 사용하기 위해 80x32비트 로컬 버퍼를 사용하고 있다. 이 방식은 YCbCr 매크로 블록에 대한 디블록킹 필터링을 수행하는데 192(Y)+72(Cb)+ 72(Cr) = 336 클럭 사이클을 소요한다.

논문 [6]은 64x32비트의 RAM 1개와 96x32비트 RAM 2개를 사용하여 구현하였다. 이는 4×4 블록의 입력 순서를 조정하여 수평 방향 필터링이 완료되면 바로 수직 방향 필터링을 수행할 수 있도록 하였다. 그러나 YCbCr 매크로 블록을 처리하는데 446 클럭 사이클을 소요되어 논문 [5]보다 성능 떨어지며, 많은 메모리를 사용하는 단점이 있다.

논문 [8]은 1차원 필터 기반의 이중 포트 메모리를 사용한 디블록킹 필터를 제안하였다. 160×32비트 이중 포트 메모리를 사용하여 구현하였으나, 매크로 블록에 대한 디블록킹 필터링을 수행하는데 814 클럭 사이클이 소요되어 속도가 느리며, 메모리 사용 또한 많은 것이 단점이다.

여러 논문을 비교한 결과 1-D 필터를 사용하는 것보다 2-D를 사용하여 처리할 경우 빠른 처리 속도를 보일 수 있지만 메모리를 많이 사용하는 단점이 있다. 그리고 1-D 필터를 사용할 경우 2-D 필터 보다 지연이 많이 발생하지만 [3]논문과 같이 적절한 메모리 사용으로 처리 속도도 향상시키고 메모리도 적게 사용할 수 있다.

이에 본 논문은 [3]논문의 장점과 2-D 필터의 장점을 이용하여 [3]논문보다 처리 속도도 향상시키고 메모리도 적게 사용하는 디블록킹 필터의 아키텍처를 연구하였으며, 블록내의 파이프라인 뿐만 아니라 블록간의 파이프라인 처리 구현을 연구하여 성능 향상을 시켰다.

Ⅲ. 디블록킹 필터

H.264/AVC는 4×4 블록을 기본으로 DCT와 양자화를 통한 변환을 수행하고, 움직임 보상에 대해서도 역시 4×4의 작은 블록 크기를 사용하고 있다. 이러한 이유로 디블록킹 필터는 모든 4×4 블록 에지에 적용된다.

본 논문에서는 매크로블록(MB)이 16×16 luminance 블록과 2개의 8×8 chrominance 블록(Cr 및 Cb)으로 구성되는 4:2:0 비디오 포맷에 대한 디블록킹 알고리즘을 기반으로 설명할 것이다. H.264/AVC에서 디블록킹 필터는 매크로블록 기반으로 4×4 휘도 블록 및 4×4 색차 블록 에지에 인가된다. 각 매크로블록에 대해 수직 에지(Vertical Edge)가 먼저 왼쪽에서 오른쪽으로 필터링된 후, 수평 에지(Horizontal Edge)가 위에서 아래 방향으로 필터링된다. 그림 1에 번호로 표시된 것과 같이 먼저 Y 블록은 1,2,3,4 순서로 수직 에지에 대한 디블록킹 필터링을 수행한 다음 5,6,7,8과 같은 순서로 수평 에지에 대한 디블록킹 필터링 수행한다. Cb 및 Cr 블록도 Y블록과 동일한 방식으로 디블록킹 필터링을 수행한다.

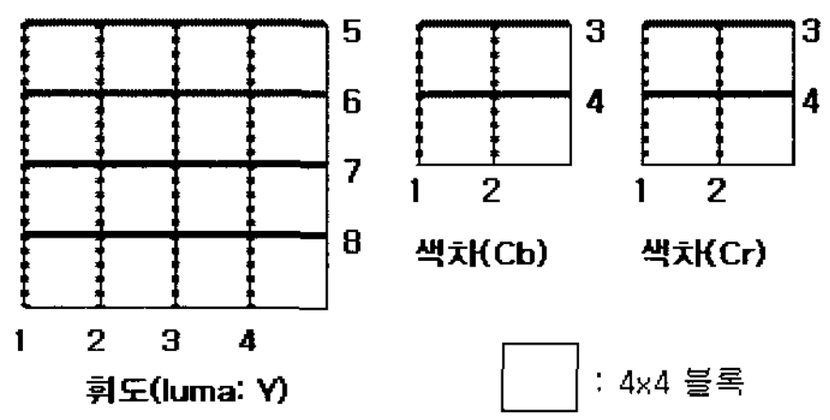


그림 1. 16x16 매크로블록에서 에지 필터 순서
Fig. 1 Edges filter order in 16x16 MB

그림 2는 휘도 매크로블록에서 수직 경계와 수평 경계에 대한 디블록킹 필터 처리 방식을 보여주고 있다. 그림에서 작은 원은 하나의 화소를 의미하며, 인접한 4×4 블록간에 라인단위로 디블록킹을 수행하게 된다. p0, p1, p2, p3 및 q0, q1, q2, q3은 필터링을 수행하지 않은 원래의 화소 값을 의미하며, P0, P1, P2, P3, Q0, Q1, Q2, Q3 는 디블록킹 필터링 처리가 완료된 후의 화소 값을 나타낸다.

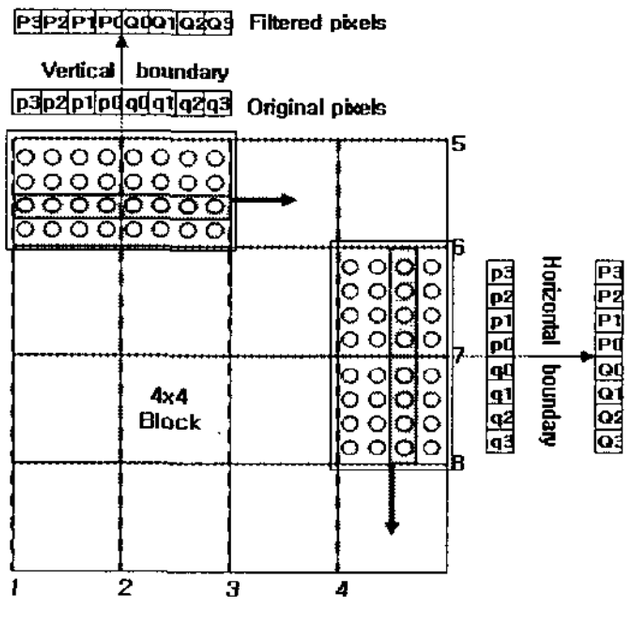


그림 2. 휘도 MB에서 4x4 블록 에지의 필터링 예
Fig.2 Example for filtering the edges of 4x4 blocks in a luminance MB

디블록킹 필터의 동작은 Bs(경계강도: Boundary Strength) 값에 따라 5개의 동작 모드가 존재한다. Bs는 해당 블록의 코딩 모드, 블록 경계의 화소 값의 차 및 블록 경계의 유형에 따라 Bs=0부터 Bs=4까지 존재한다. Bs=0인 경우에는 디블록킹 필터링을 수행하지 않는 모드이며, Bs=1부터 Bs=3까지의 모드는 표준 필터링 모드로 파라미터에 따라 0개, 2개 혹은 4개의 화소값을 필터링하게 된다. 그리고, Bs=4는 가장 강한 경계 강도 모드로 파라미터에 따라 0개, 2개 혹은 6개의 화소값을 필터링하게 된다. 그림 3은 Bs 값의 할당 과정을 보여주고 있다.

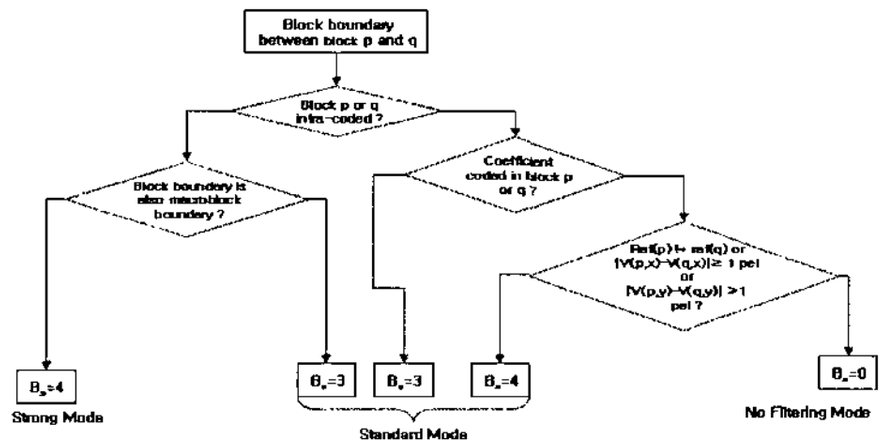


그림 3. Bs 값의 할당
Fig. 3 Assignment of boundary strength(Bs)

$B_s = 0$ 인 경우

$P_0 = p_0, P_1 = p_1, P_2 = p_2, P_3 = p_3, Q_0 = q_0, Q_1 = q_1, Q_2 = q_2, Q_3 = q_3$

$B_s = 1$ or $B_s = 2$ or $B_s = 3$ 인 경우

if $b_0 = |p_0 - q_0| \geq \alpha$ or $b_p = |p_1 - p_0| \geq \beta$ or $b_q = |q_1 - q_0| \geq \beta$ *then*
$P_0 = p_0, P_1 = p_1, P_2 = p_2, P_3 = p_3, Q_0 = q_0, Q_1 = q_1, Q_2 = q_2, Q_3 = q_3$
else
if $a_p = |p_2 - p_0| < \beta$ *then*
$t_c = t_{co} + 1$
$P_1 = p_1 + dip3(-t_{co}, t_{co}, (p_2 + (p_0 + q_0 + 1) \gg 1 - (p_1 \ll 1)) \gg 1)$
else
$P_1 = p_1, t_c = t_{co}$
$\Delta = dip3(-t_c, t_c, (((q_0 - p_0) \ll 2 + ((p_1 - q_1) + 4 \gg 3)$
$P_0 = dip1(p_0 + \Delta), P_2 = p2$
if $a_q = |q_2 - q_0| < \beta$ *then*
$Q_1 = q_1 1 + dip3(-t_{co}, t_{co}, (q_2 + (p_0 + q_0 + 1) \gg 1 - (q_1 \ll 1)) \gg 1),$
$t_c = t_{co} + 1$
else
$Q_1 = q_1, t_c = t_{co}$
$\Delta = dip3(-t_c, t_c, (((q_0 - p_0) \ll 2 + ((p_1 - q_1) + 4) \gg 3)$
$Q_0 = dip1(q_0 - \Delta), Q_2 = q_2, Q_3 - q_3$
where $dip3(A, B, K) = A$ if $K < A, B$ if $K > B$, *else* K
$dip1(A) = 0$ if $A < 0, 255$ if $A > 255$, *else* A

$B_s = 4$ 인 경우

if $b_0 = |p_0 - q_0| \geq \alpha$ or $b_p = |p_1 - p_0| \geq \beta$ or $b_q = |q_1 - q_0| \geq \beta$ *then*
$P_0 = p_0, P_1 = p_1, P_2 = p_2, P_3 = p_3, Q_0 = q_0, Q_1 = q_1, Q_2 = q_2, Q_3 = q_3$
else
if $a_p = |p_2 - p_0| < \beta$ and $b_0 < ((\alpha \gg 2) + 2)$ *then*
$P_0 = (p_2 + (p_1 \ll 1) + (p_0 \ll 1) + (q_0 \ll 1) + q_1 + 4) \gg 3$
$P_1 = (p_2 + p_1 + p_0 + q_0 + 2) \gg 2$
$P_2 = ((p_3 \ll 1) + p_2 + (p_2 \ll 1) + p_1 + p_0 + q_0 + 4 \gg 3, P_3 = p_3$
else
$P_0 = (p_1 \ll 1) + p_0 + p_1 + 2) \gg 2, P_1 = p_1, P_2 = p_2, P_3 = p_3$
if $a_q = |q_2 - q_0| \ll \beta$ and $b_0 < ((\alpha \gg 2 + 2)$ *then*
$Q_0 = (p_1 + (p_0 \ll 1) + (q_0 \ll 1) + (q_0 \ll 1) + q_2 + 4) \gg 3$
$Q_1 = (p_1 + q_0 + q_1 + q_2 + 2) \gg 2$
$Q_2 = ((q_3 \ll 1) + q_2 + (q_2 \ll 1) + q_1 + p_0 + q_0 + 4 \gg 3, Q_3 = q_3$
else
$Q_0 = (q_1 \ll 1) + q_0 + p_1 + 2) \gg 2, Q_1 = q_1, Q_2 = q_2, Q_3 = q3$

그림 4. 디블록킹 필터링 과정
Fig. 4 Process of deblocking filtering

이 밖에도 디블록킹 필터링 연산을 수행할 때에는 양자화 파라미터 QP 값에 영향을 받는 임계값 α와 β가 사용되고, QP 및 Bs 값 모두에 영향을 받는 클리핑 파라미터인 t_{co} 값이 사용된다[9]. 이러한 필터를 사용한 디블록킹 처리는 그림 4에 자세히 수식으로 설명하였다. 그림 5는 QP 값의 변화에 따른 α 및 β 값의 설정을 보여주고 있으며, 그림 6은 QP 및 Bs 값의 변화에 대응하는 클리핑 파라미터 t_{co} 값의 설정을 보여주고 있다.

QP	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
α	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	4
β	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2
QP	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
α	5	6	7	8	9	10	12	13	15	17	20	22	25	28	32	36	40	45
β	2	3	3	3	3	4	4	4	6	6	7	7	8	8	9	9	10	10
QP	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51		
α	50	56	63	71	80	90	101	113	127	144	162	182	203	226	255	255		
β	11	11	12	12	13	13	14	14	15	15	16	16	17	17	18	18		

그림 5. QP에 따른 α 및 β 파라미터
Fig. 5 α and β parameters according QP values

QP	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$B_s=1$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$B_s=2$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$B_s=3$ or 4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
QP	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
$B_s=1$	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	2	2	2
$B_s=2$	0	0	0	1	1	1	1	1	1	1	1	1	1	2	2	2	2	3
$B_s=3$ or 4	1	1	1	1	1	1	1	1	1	2	2	2	2	3	3	3	4	4
QP	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51		
$B_s=1$	2	3	3	3	4	4	4	5	6	6	7	8	9	10	11	13		
$B_s=2$	3	3	4	4	5	5	6	7	8	8	10	11	12	13	15	17		
$B_s=3$ or 4	4	5	6	6	7	8	9	10	11	13	14	16	18	20	23	25		

그림 6. 클리핑 파라미터 t_{co}
Fig. 6 Clipping parameters t_{co}

Ⅳ. 디블록킹 필터 설계

4.1 디블록킹 필터의 처리 순서 정의

서론에서 이미 언급한 바와 같이 디블록킹 필터는 H.264/AVC 디코더에서 복호 연산량의 1/3을 점유할 정도로 연산량이 많은 블록이다. 따라서 4×4 블록들의 입력 순서 및 디블록킹 필터링의 처리 순서는 디블록킹 필터의 성능에 중대한 영향을 미친다. 그림 7은 디블록킹 필터 연산을 수행하기 위한 4×4 블록의 입력 순서를 보여주고 있다. 1번 블록 입력을 시작으로 오름차순으로 연속적으로 입력되어 40번 블록까지 입력되면 하나의 매크로블록에 대한 처리를 완료하게 된다. 여기서 회색 영역으로 표시된 블록들은 이웃한 매크로블록의 영역을 의미한다. 즉, 하나의 매크로블록에 대한 디블록킹 필터링을 수행할 경우에는 왼쪽 경계의 매크로블록 일부분과 위쪽 경계의 매크로블록 일부분이 사용된다.

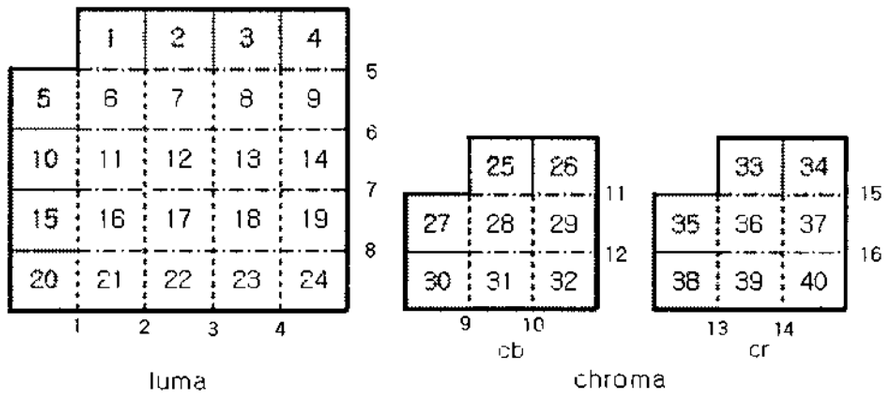
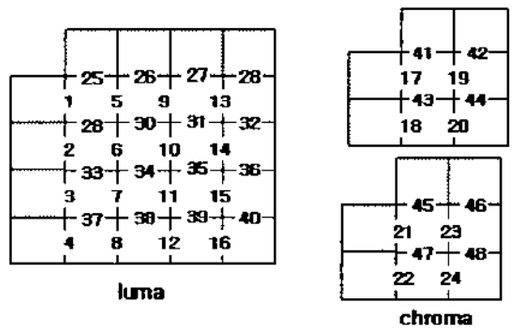
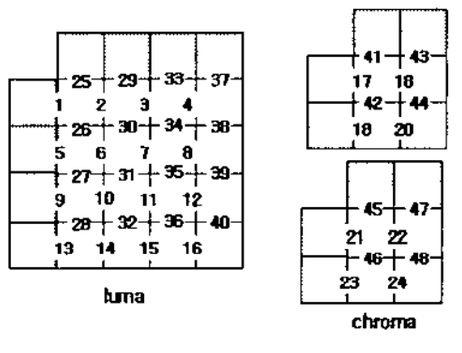


그림 7. 디블록킹 필터의 입력 순서
Fig. 7 Input order of deblocking filter

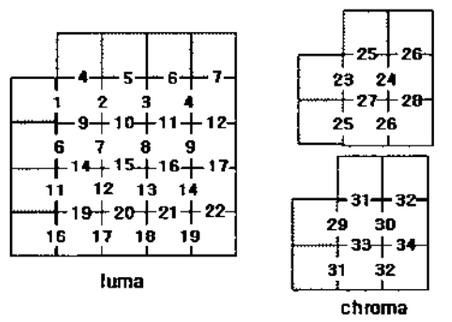
디블록킹 필터에 대한 처리 순서는 수직 에지에 대한 수평 필터링을 항상 먼저 수행한 이후에 수평 에지에 대한 수직 필터링을 수행하도록 정의되어 있다. 그림 8의 (a)는 디블록킹 필터의 기본적인 처리 순서를 보여주고 있다[8]. 논문 [5]에서는 그림 (b)는 디블록킹 필터 처리를 위한 향상된 1차원 처리 순서를 제안하여 구현하였다. 그러나, 그림(a)와 그림(b)는 내부 메모리 사용이 많고 필터 처리 시간이 많이 소요되는 단점이 있다.



(a) 기본적인 필터 처리 순서 (a) Basic processing order



(b) 향상된 1차원 처리 순서[5] (b) Advanced 1-D processing order



(c) 2차원 처리 순서[6] (c) 2-D processing order

그림 8. 디블록킹 필터 처리 순서
Fig. 8 Deblocking filter processing orders

그림 8의 (c)는 논문 [6]에서 제안한 방식을 본 논문에서 필터의 지연을 고려하여 수정한 순서로 내부 버퍼의 사용을 줄이면서 외부 메모리 액세스를 줄일 수 있는 장점이 있다.

따라서 본 논문에서 2차원 필터 처리 순서를 사용하면서도 기존의 논문 [6]과 비교하여 월등한 성능을 지닐 수 있는 새로운 하드웨어 구조를 제안하고자 한다. 아울러 논문 [8]에서 제안한 알고리즘을 한층 개선하여 성능 향상을 도모하였다.

4.2 제안하는 디블록킹 필터의 설계

H.264/AVC에서 디블록킹 필터는 블록 기반의 변환 및 양자화를 통하여 발생되는 블록화 현상을 제거하기 위해 부호기 및 복호기에 루프 필터 형태로 존재한다. 표준안에 따르면 16×16의 휘도 성분과 8×8의 색차 성분을 수평 방향으로 필터 처리한 후 수직 방향으로 필터 처리를 한다. 양자화 레벨에 따른 여러 가지 파라미터들을 통하여 필터 처리의 유무를 선택하며, 필터 처리 시 크게 덧셈, 쉬프트 및 클리핑 처리 등으로 구성된다. 실시간 처리를 위한 하드웨어 설계 시 필터 처리를 하려면 표준안에서 규정된 처리 방법과 결과가 동일하면서도 빠른 속도로 동작을 해야 한다.

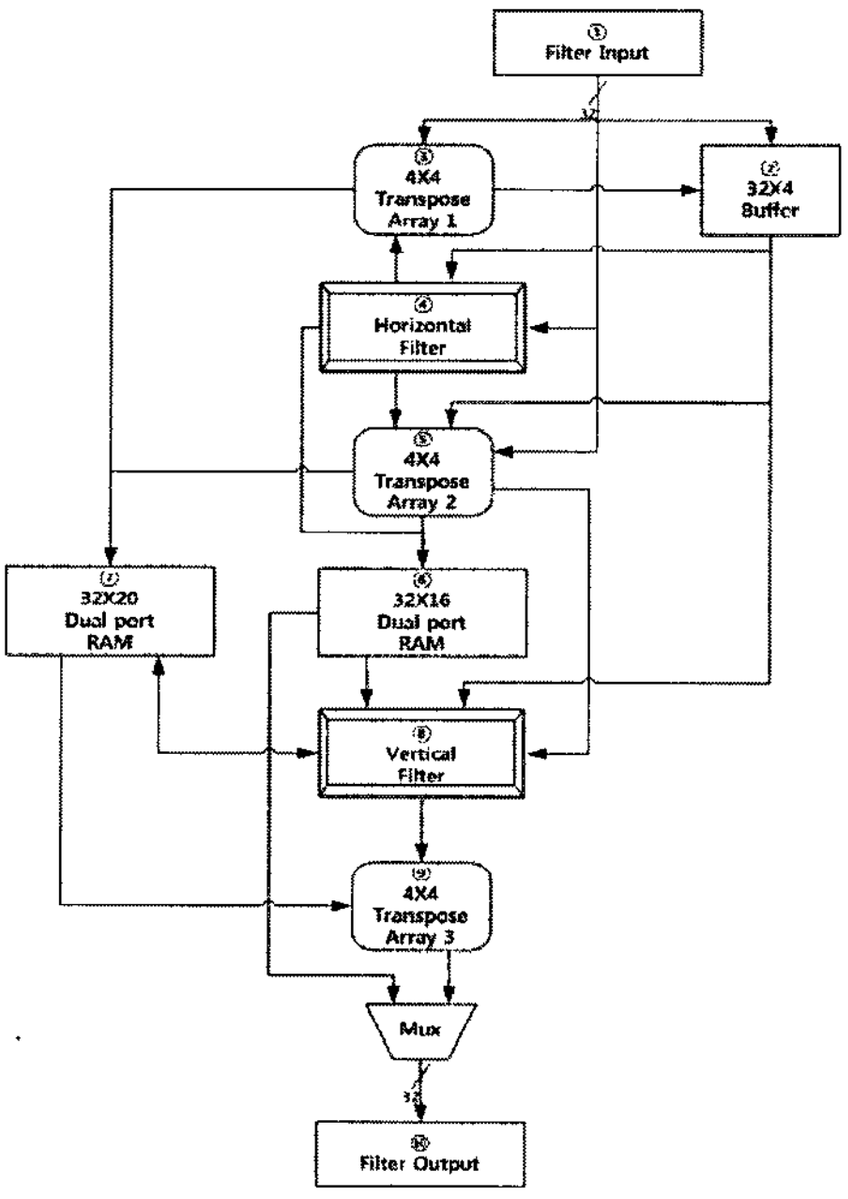


그림 9. 제안하는 디블록킹 필터의 아키텍처
Fig. 9 Architecture for the proposed deblocking filter

그림 9는 본 논문에서 제안하는 디블록킹 필터의 아키텍처를 보여주고 있다. 제안하는 방법은 수평 필터와 수직 필터 2개를 사용하여 수평 필터 처리가 완료된 4×4 블록을 전치(Transpose) 회로를 통하여 수직 필터의 입력 형식으로 변환한 후 곧바로 수직 필터 처리를 수행한다.

이것은 수평 필터 처리가 모두 끝나기 전에 곧바로 수직 필터 처리가 가능한 블록을 수행함으로써 한 프레임의 전체 수행 사이클 수를 줄일 있도록 하며, 사용하는 메모리의 크기를 줄일 수 있는 장점을 제공한다.

이 방법의 구현을 위해서는 입력 데이터는 16×16 휘도 블록과 인접한 상위 4개의 4×4 블록, 좌측 4개의 4×4 블록 및 8×8 색차 블록과 인접한 상위 2개의 4×4 블록, 좌측 2개의 4×4 블록으로 총 40개의 4×4 블록을 입력받게 되며 한 클럭당 4개의 픽셀 데이터(32비트)를 수신한다.

수평 필터 처리 후 수직 필터를 처리하기 위하여 D 플립플롭의 배열로 구성된 전치 배열(Transpose Array) 아키텍처를 사용하였는데, 이는 시간 지연없이 데이터를 정렬해주는 기능을 수행하는 블록이다. 이에 대한 것은 그림 10에 자세히 나타나 있다. 여기서 sw 신호는 매 4 클럭마다 토글링하게 되며, 이에 따라 4 클럭단위로 전치된 픽셀 데이터를 출력하게 된다[6].

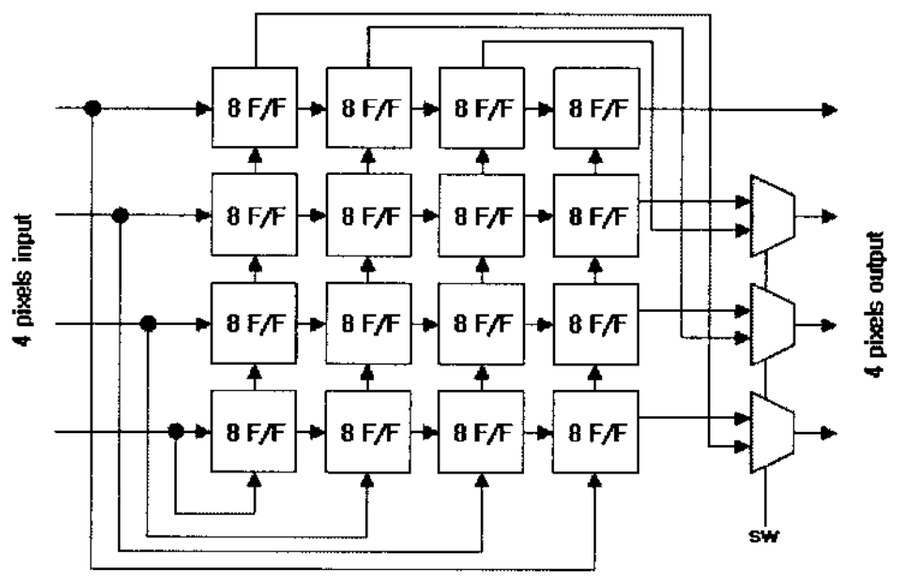


그림 10. 4x4 전치 배열 아키텍처
Fig. 10 A 4x4 transpose array architecture

현재 블록이 수평필터 처리를 위해 다음 블록이 입력으로 올 때까지 저장되어야 할 32×4 버퍼가 필요하며, 현재 수행하려는 매크로 블록의 상위 4×4 블록들은 수평 필터 처리를 할 필요가 없기 때문에 수직 필터 처리를 위해 저장되거나, 수평 필터 처리 후 수직 필터 처리까지 저장되어야 할 버퍼가 필요한데 이는 30×20 이중 포트 RAM으로 구현하였다. 마지막으로 좌측의 4×4 블록들은 한 번의 수평 필터 처리 후 수직 필터 처리과정은 필

요 없기 때문에 출력 순서를 맞춰주어야 하며 위의 30×20 RAM에서 이중 포트가 모드 사용되어지고 있을 때 임시적인 저장이 필요하여 32×16크기의 이중 포트 RAM으로 구현하였다.

제안하여 설계된 디블록킹 필터의 시간 스텝별 처리 과정이 그림 11에 자세히 나타나 있다.

Time step	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
4x4 input	1	2	3	4	5	6	7	8	9	10	11
HF input						5-6	6-7	7-8	8-9		10-11
VF input									1-6	2-7	3-8
Time step	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21
4x4 input	12	13	14	15	16	17	18	19	20	21	22
HF input	11-12	12-13	13-14		15-16	16-17	17-18	18-19		20-21	21-22
VF input	4-9		5-11	7-12	8-13	9-14		11-16	12-17	13-18	14-19
Time step	S22	S23	S24	S25	S26	S27	S28	S29	S30	S31	S32
4x4 input	23	24			25	26	27	28	29	30	31
HF input	22-23	23-24						27-28	28-29		30-31
VF input		16-21	17-22	18-23	19-24						25-28
Time step	S33	S34	S35	S36	S37	S38	S39	S40	S41	S42	S0'
4x4 input	32	33	34	35	36	37	38	39	40		1
HF input	31-32				35-36	36-37		38-39	39-40		
VF input	26-29		28-31	29-32				33-36	34-37		36-39

Where
Time step : 각 상태는 4클럭 사이클 소요(S0 ~ S43)
HF : Horizontal Filtering
VF : Vertical Filtering
Number 1 ~ 40 : 각 숫자는 4x4 블록 데이터
A-B : 블록 A와 블록 B 사이의 필터링 수행한 의미
최소 출력 시점 : S10 시간 스텝에서 1번의 4x4블록 출력 시작

휘도성분의 필터링이 끝나는 지점 (S0 to S26)

그림 11. 디블록킹 필터의 시간 단계별 과정
Fig. 11 Time step flow for the designed deblocking filter

그림 11에서 시간 스텝 $s_i(i=0,1,2,...)$는 하나의 4×4 블록을 처리하는데 소요되는 시간 단위로 4클럭 사이클로 동일하다. 시간 스텝 s_0에서 s_3 동안 디블록킹 수행을 위한 초기화 단계로 블록 1번부터 블록 4번까지를 전치 배열 스위치로 입력하여 수직 필터링을 위한 입력 형태로 변환하며, 5번 블록을 32×16 버퍼에 저장을 한 후, 시간 스텝 s_5에서 최초로 블록 5번과 블록 6번 사이의 수평 필터링을 수행한다. 그리고 시간 스텝 s_8부터 수평 필터링과 이미 수평 필터링이 완료된 블록에 대한 수직 필터링을 수행하여 동일한 시간 스텝에 수평 및 수직 필터링을 수행하게 된다. 시간 스텝 s_{10}부터 입력 순서와 동일한 순서로 블록 1번부터 필터링이 완료된 블록을 출력하게 된다. 그리고 최종적으로 모든 필터링을 완료한 마지막 블록인 40번 블록은 s_{49} 시간 스텝에 출력되게 된다. 이 구조는 첫 입력에서 출력까지 200 클록 사이클(50 시간스텝 × 4사이클/스텝= 200)이 소요되지만 출력이 완료되기 전에 순차적으로 매크로 블록이 입력 가능하기 때문에 첫 10 시간 스텝의 출력 지연을 제외하고는 43 시간 스텝 마다 하나의 매크로블록이 출력되어지므로 본 논문에서 제안한 디블록킹 필터에서 모든 필터링 처리를 마치고, 외부로 출력을 완료하는데 까지 걸리는 시간은 172 클록 사이클(43 시간 스텝 × 4사이클/스텝 = 172)이 소요된다.

본 논문에서 제안한 방식의 디블록킹 필터링의 특징은 수평 필터와 수직 필터를 동시에 수행할 수 있는 하드웨어 자원을 제공하여 수평 필터링과 수직 필터링을 동시에 수행함으로써 디블록킹 필터 처리의 전체 수행 시간을 줄였다.

V. 설계 결과 고찰

본 논문에서 설계한 디블록킹 필터의 수행 사이클을 3개의 매크로블록이 입력되었을 때를 요약하여 표현하면 그림 12와 같으며, 지속적으로 매크로블록이 입력 될 수 있다. 그림 9의 디블록킹 필터 아키텍처를 통해 그림 12의 동작과정을 설명하면 다음과 같다.

휘도와 색차 순으로 32bit(8bit×4pixel)씩 4클록에 하나의 4×4 블록이 입력되어 수평 필터, 전치 배열, 버퍼에 전달된다. 입력 모듈 ①에서는 수평 필터 ④에 동시에 입력되어지는 다음 시간 스텝의 4×4블록과 동기를 맞추기 위해 버퍼 모듈 ②로 전달되고, 전치 배열 모듈 ③과 ⑤에는 수평 필터링이 수행된 블록이나 수평 필터링이 필요 없는 블록을 수직 필터링을 위해 전치를 수행한다. 여기서, 전치 배열이 2개가 존재하는 이유는 수평 필터가 수행되면 동시에 2개의 블록이 출력되는데 이를 지연 없이 수직 필터링을 하기 위함이다. 입력 모듈 ①에서 현재 입력된 4×4 블록과 버퍼 모듈 ②에 저장된 이전 4×4블록이 수평 필터링을 위해 수평 필터 모듈 ④로 곧바로 입력되어진다.

버퍼 모듈 ②는 입력 모듈 ①이나 전치배열 모듈 ③에서 전달되어진 블록을 다음과 같은 목적에 따라 다음 시간 스텝으로 지연시킨다. 버퍼 모듈 ②에서 수평 필터 ④와 수직 필터 ⑧로의 전달 과정은 입력 모듈 ①에서 하나의 블록씩 순차적으로 들어오지만 필터의 입력은 동시에 두 개의 블록이 들어가므로 이전 시간 스텝에 입력된 블록은 버퍼에 저장하여 다음 시간 스텝에 수평 필터링 또는 수직 필터링을 수행하기 위함이고, 전치 배열 모듈 ⑤로는 그림 4에 25, 26, 33, 34번 블록이 이전 휘도의 수직 필터링이 끝나야 입력되어질 수 있기 때문에 지연시키기 위해 임시로 저장한다.

전치 배열 모듈 ③과 ⑤는 수직 필터링을 위해 전치를 수행하고 RAM ⑥과 ⑦또는 버퍼 모듈 ②에 저장하거나 수직 필터 ⑧에 전달한다.

수평 필터 ④의 출력은 수직 필터링을 위해 전치 배열 모

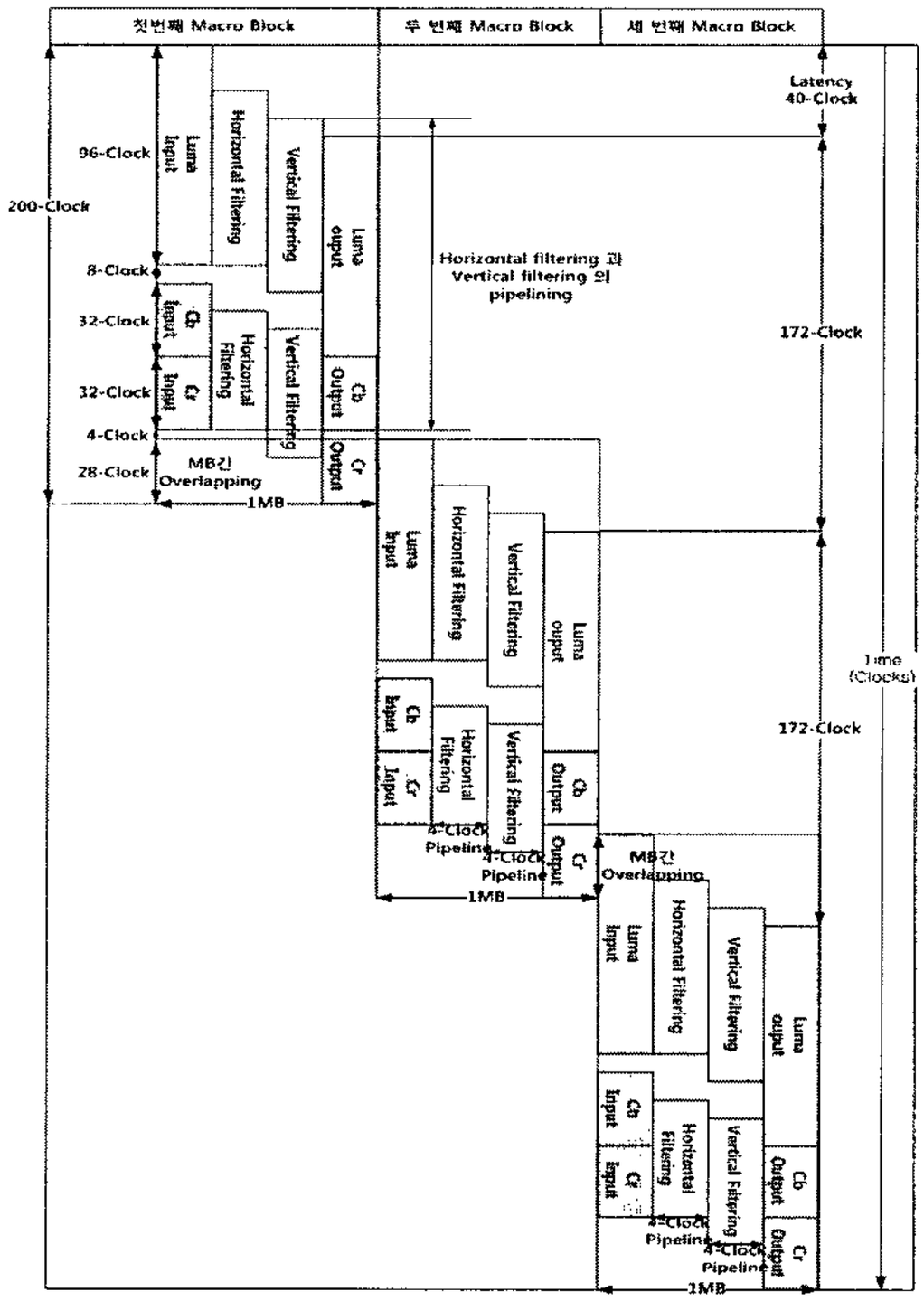


그림 12. 디블록킹 필터의 수행 사이클
Fig. 12 Execution cycles for the deblocking filter

듈 ③ 또는 ⑤에 전달되어 전치를 수행하거나 수직 필터링 수행이 필요 없는 블록(그림4의 5, 10, 15, 20, 27, 30, 35, 38 번 블록)을 32×16 이중 포트 RAM ⑥에 저장하였다가 출력 시점에 맞춰 나가게 된다.

32×16 이중 포트 RAM ⑥은 수직 필터링 수행이 필요 없는 블록을 출력으로 전달하거나 32×20 이중 포트 RAM ⑦의 이중 포트가 모두 사용되어지고 있을 때 저장하여 수직 필터 ⑧로 전달한다.

32×20 이중 포트 RAM ⑦은 저장되어있는 블록을 수직 필터링 시점에 맞추어 모듈 ⑧에 전달하고 출력 시점에 맞추어 전치 배열 모듈 ⑨에 전달한다.

수직 필터 ⑧은 필터링을 수행하여 블록을 출력 순서에 따라 32×20 이중포트 RAM ⑦ 또는 전치 배열 모듈 ⑨에 전달한다.

전치 배열 모듈 ⑨는 수직 필터링이 수행된 블록들을 출력하기 위해 원래대로 전치를 수행한다.

블록이 처음 입력되어지는 시점에서 40-Clock 이후에 Filtering이 끝난 1번 블록부터 40번 블록까지 순차적으로 출력한다.

이와 같이 입력되는 픽셀 데이터의 순서를 적절히 조작하여 수평 필터링과 수직 필터링을 동시에 수행하게 하고, 매크로블록간의 겹침(Overlapping)을 통하여 하나의 매크로블록의 출력이 완료되지 않았더라도 입력이 정해져 있는 순서에 따라 들어가게 되어 전체적인 처리 시간을 172 클록 사이클로 줄였다.

본 논문에서 제안하는 디블록킹 필터 아키텍처에 대해 기존에 발표된 다른 논문들과 비교를 수행하였다. 표 1은 본 논문에서 제안한 아키텍처와 기존에 발표된 아키텍처 사이의 몇 가지 중요한 요소에 대해 비교를 수행한 것이다.

표 1. 아키텍처 비교
Table 1. Architecture comparison

구분	논문 [5]	논문 [6]	논문 [8]	The proposed architecture
clk/MB	336	446	814	172
Dual port memory	×	160x32	160x32	36x32
Single port memory	80x32	×	×	⁄

표1을 분석해 보면 알 수 있듯이 본 논문에서 제안한 디블록킹 필터 아키텍처를 기존의 논문들과 비교하여 볼 때, 적은 양의 메모리를 사용하면서 디블록킹 필터의 처리 순서에 대한 분석을 토대로 수평 필터링과 수직 필터링을 병렬로 수행함으로써 매크로블록에 대한 디블록킹 필터링의 수행 사이클을 기존의 논문과 비교하여 1.95배에서 4.73배까지 향상시켰다. 이처럼 더 적은 양의 메모리를 사용하고, 매크로블록당 수행 사이클을 대폭 줄임으로서 2048×1024와 같은 높은 해상도를 필요로 하는 비디오 응용에서 실시간 디블록킹 필터링을 수행할 수 있어 향후 적용이 가능할 것으로 예견되는 바이다.

VI. 결 론

디블록킹 필터는 H.264/AVC 디코더에서 복호 연산량의 1/3을 점유할 정도로 연산량이 많은 블록이다. 표준안에 따르면 16×16의 휘도 성분과 8×8의 색차 성분을 수평 방향으로 필터 처리한 후 수직 방향으로 필터 처리를 한다. 실시간 처리를 위한 하드웨어 설계 시 필터 처리를 하려면 표준안에서 규정된 처리 방법과 결과가 동일하면서도 빠른 속도로 동작을 해야 한다. 본 논문에서는 이

러한 규정을 준수하면서 수평 필터링과 수직 필터링을 동시에 수행할 수 있는 아키텍처를 제안하였다. 제안한 디블록킹 필터 아키텍처는 기존의 논문과 비교하여 성능이 1.95배에서 4.73배까지 향상되었으며, 메모리 사용량 또한 감소시켰다.

본 논문에서 제안한 디블록킹 아키텍처는 2048×1024와 같이 높은 해상도를 필요로 하는 비디오 응용에서 실시간 디블록킹 필터링을 수행할 수 있을 것으로 판단되는 바 향후 고품질의 DMB와 PMP 등에 적용이 가능할 것으로 예상된다.

참고문헌

[1] ITU-T Rec. H.264, "Advanced Video Coding for Generic Audio Visual Services", 2005

[2] Lain E.G. Richardson, "H.264 and MPEG-4", 홍릉과학출판사, 2004

[3] T. Wiegand, G. J. Sullivan, and A. Luthra, "Overview of the H.264/AVC Video Coding Standard", IEEE Trans. on Circuits and Systems for Video Technology, Vol.13, No.7, pp560-576, July 2003

[4] Miao Sima, Yuanhua Zhou, and Wei Zhang, "An Efficient Architecture for Adaptive Deblocking Filter of H.264/AVC Video Coding", IEEE Trans. on Consumer Electronics, Vol.50, No.1, pp292-296, Feb. 2004

[5] Chao-Chung Cheng, Tian-Sheuan Chang, "An Hardware Efficient Deblocking Filter for H.264/AVC", International Conference on Comsumer Electronics. pp235-236, Jan. 2005

[6] Bin Sheng, Wen Gao, Di Wu, " An Implemented architecture of deblocking filter for H.264/AVC", IEEE 2004 ICIP, Vol.1, pp665-668, Oct. 2004.

[7] 손승일, 김원삼, "H.264/AVC용 병렬 디블록킹 필터의 아키텍처에 관한 연구", 한국해양정보통신학회 논문지, 제11권 제4호, pp.765-772, Apr. 2007.

[8] Y.-W. Huang, T.-W. Chen, B.-Y. Hsieh, T.-C.Wang, T.-H. Chang, and L.-G. Chen, "Architecture design for deblocking filter in H.264/JVT/AVC", IEEE International Conference on Multimedia and Expo (ICME 2003), pp693-696, Jul. 2003.

[9] Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification(ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC), Geneva, Switzerland, May. 2003.

저자소개

김 원 삼(Won-Sam Kim)

2008년 한신대학교 컴퓨터정보학과 (석사)

※관심분야 : ASIC 설계, H.264/AVC 코덱

손 승 일(Seung-Il Sonh)

1989년 연세대학교 전자공학과 (공학사)
1991년 연세대학교 대학원 전자공학과 (공학석사)
1998년 연세대학교 대학원 전자공학과(공학박사)
2002년~현재 한신대학교 정보통신학과 부교수
※관심분야 : ATM 통신 및 보안, ASIC 설계, 영상신호처리칩, 비디오코덱