논문 2008-45SD-7-8

H.264/AVC를 위한 디블록킹 필터의 효율적인 VLSI 구조

(An Efficient VLSI Architecture of Deblocking Filter in H.264 Advanced Video Coding)

이 성 만*, 박 태 근**

(Sungman Lee and Taegeun Park)

요 약

디블록킹 필터는 H.264/AVC의 디코딩 과정에서 생기는 블록 왜곡 현상을 없애주고 압축율을 높여준다. 하지만 디블록킹 필터는 디코더에서 1/3의 계산 량을 차지할 만큼 계산 량이 많아 이를 위한 효율적인 하드웨어 설계가 필요하다. 본 논문에서는 적절한 메모리 구조를 사용하여 데이터의 재사용을 높이고, 두 개의 필터를 사용하여 성능을 개선한 디블록킹 필터의 구조를 제안한다. 제안된 구조는 적은 초기화 클럭 이후 두 개의 필터가 동시에 동작하여 데이터가 준비되는 대로 필터링을 수행하여 처리량을 높이고, 외부메모리의 참조를 최소화한다. 제안된 구조는 하나의 매크로블록을 필터링하는 데에 96클럭이 소요되며. 동부아남 0.18㎛ 표준 셀 라이브러리를 사용하여 합성한 결과 최대 동작 주파수는 200㎒이다.

Abstract

The deblocking filter in the H.264/AVC video coding standard helps to reduce the blocking artifacts produced in the decoding process. But it consumes one third of the computational complexity in H.624/AVC decoder, which advocates an efficient design of a hardware accelerator for filtering. This paper proposes an architecture of deblocking filter using two filters and some registers for data reuse. Our architecture improves the throughput and minimize the number of external memory access by increasing data reuse. After initialization, two filters are able to perform filtering operation simultaneously. It takes only 96 clocks to complete filtering for one macroblock. We design and synthesis our architecture using Dongbuanam 0.18㎛ standard cell library and the maximum clock frequency is 200㎒.

Keywords: H.264/AVC, deblocking filter, VLSI architecture

I. 서 론

새로운 통신환경이 등장하면서 더 뛰어난 압축효율을 제공하며 통신채널에 알맞은 비디오 압축 표준이 필요하게 되었다. Moving Picture Experts Group과 Video Coding Experts Group은 MPEG-4와 H.263보다 압축율이 향상된 H.264/AVC를 발표하였다[1].

H.264/AVC는 가변크기 블록단위의 움직임 예측 및 보상, 인-루프 디블록킹 필터, 향상된 엔트로피 부호화, 다양한 네트워크에 적응하기 위한 NAL(Network Abstraction Layer) 등의 사용으로 압축율이 향상되었지만, 블록기반 코딩기법을 사용하는 H.263, MPEG-2, MPEG-4등과 같이 블록 왜곡 현상이 발생된다. 그림 1은 H.264/AVC 인코더의 구조를 나타낸 것이다.

인코더는 영상을 매크로블록 단위로 압축을 하여 코딩하는 부분과 다시 영상을 복원하는 디코더 부분을 포함하고 있다. 디블록킹 필터는 디코더 부분에 위치하여 복원영상이 원영상에 더 가깝도록 도움으로써 영상간의 오차값을 줄여 압축률 향상에 기여하고, 디코더에서는

* 학생회원, ** 정회원, 가톨릭대학교 정보통신전자공학부
(School of Information, Communication and Electronics Engineering, The Catholic University of Korea)

※ 본 연구는 2007년도 가톨릭대학교 교비연구비의 지원으로 이루어졌음.

접수일자: 2008년2월12일, 수정완료일: 2008년6월30일

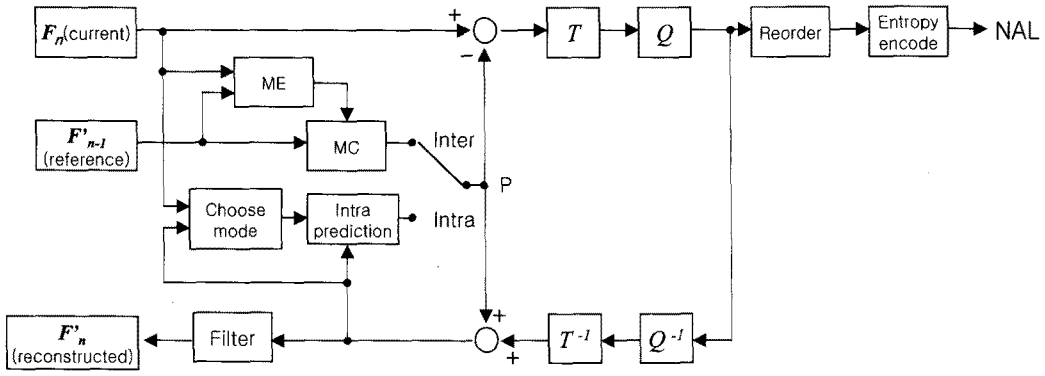

그림 1. H.264/AVC의 인코더
Fig. 1. H.264/AVC encoder.

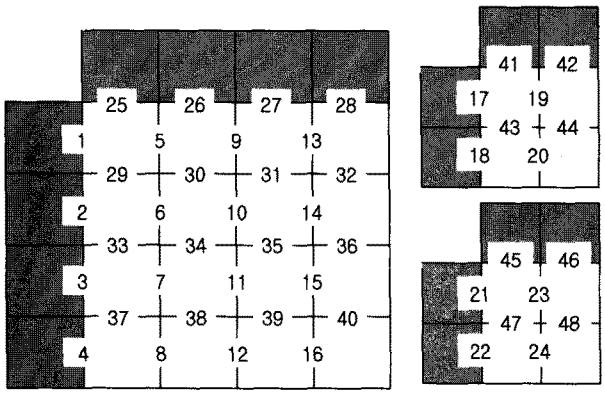

그림 2. 표준안의 필터링 순서
Fig. 2. Basic filtering order.

블록왜곡현상을 제거하여 영상의 화질을 개선한다[2].

H.264/AVC 표준에서의 디블록킹 필터는 기존의 영상압축표준에 비해 더 복잡한데, 그 이유는 첫째, H.264의 디블록킹 필터는 영상의 특성에 따라 매우 적응적이고, 둘째, 영상의 모든 매크로블록의 4x4 휘도블록과 색차블록의 경계에 적용되며, 셋째, 경계근처 3 개의 픽셀 값까지 필터링하고, 이에 따라 필요한 4x4 블록의 픽셀 값들을 읽어와야 하며 처리하고 다시 저장해야 하기 때문이다. 이와 같이 디블록킹 필터는 복잡하고 계산량이 많아 디코더 전체 계산 복잡도의 1/3을 차지한다[3~4].

이러한 문제를 해결하는 한 가지 방법은 필터링의 순서를 조절하여 데이터의 재사용을 최대한 높이는 것이다. 그림 2는 표준안의 필터링 순서를 나타낸 것으로 기본적으로 수직경계를 먼저 필터링하고, 수평경계를 필터링하며 항상 좌에서 우로, 위에서 아래로 진행된다.

표준안의 필터링 순서를 따르면 경계를 필터링할 때마다 필요한 데이터를 메모리에서 읽어 와야 하지만, 순서를 조절하여 블록의 양 옆 수직경계를 바로 연이어 필터링하면 앞 경계의 필터 출력 값이 다시 다음 경계의 필터 입력 값으로 사용할 수 있으므로 데이터의 메모리 입출력 횟수를 줄일 수 있다. 또한 필터내부에 약간의 메모리를 두어 필터링이 완전히 끝나지 않은 블록을 저장하여 두 개의 수평경계 필터링까지 끝낸 후 출력한다면 성능을 개선할 수 있다[5].

본 논문에서는 데이터의 재사용을 최대한 높이기 위해 두 개의 필터와 내부 메모리를 사용하여 성능을 개선한 디블록킹 필터의 새로운 구조와 필터링 순서를 제안한다. 본 논문의 구성은 다음과 같다. 먼저, Ⅱ장에서 디블록킹 필터의 알고리즘에 대해 설명하고, Ⅲ장에서 제안된 디블록킹 필터의 순서와 구조, 동작을 소개한다. 그리고 Ⅳ장에서 제안된 구조와 기존의 구조를 비교하며 성능을 분석하고, Ⅴ장에서 결론을 맺으며 본 논문을 마친다.

Ⅱ. 디블록킹 필터의 알고리즘

1. 필터링 결정

경계에 수행되는 필터의 세기는 경계세기(bS)와 경계주위의 이미지 샘플 값의 변화(gradient)에 따라 0에서 4까지의 값을 갖는다. 매크로블록이 인트라 코딩되거나, 인터 코딩된 경우 코딩계수를 포함하는 등 블록왜곡 현상이 일어날 가능성이 큰 경우 더 큰 bS 값을 가지며 따라서 더 강한 필터가 적용된다.

각 경계의 필터링 적용여부는 bS 값과 경계 주위 픽셀 값의 차이에 따라 정해지는데, 그 조건은 다음과 같다.

$$bS > 0 \tag{1}$$

$$|p_0 - q_0| < \alpha,\ |p_1 - p_0| < \beta,\ |q_1 - q_0| < \beta \tag{2}$$

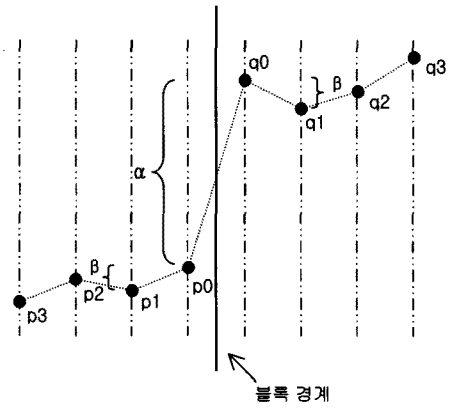

그림 3. 샘플 값과 α, β의 관계
Fig. 3. Block diagram of α, β and sample values.

α와 β는 표준안에서 정의한 임계값으로, 그림 3은 경계주위 샘플 값(p3~q3)과 α, β의 관계를 도식화한 것이다. α와 β는 식 (3)과 같이 두 블록의 양자화 파라미터(QP)값으로 구할 수 있다.

$$qP_{av} = (qP_p + qP_q + 1) \gg 1 \quad (3)$$
$$indexA = Clip(0,\ 51,\ qP_{av} + FilterOffsetA)$$
$$indexB = Clip(0,\ 51,\ qP_{av} + FilterOffsetB)$$

FilterOffsetA, B는 사용자 정의 값으로, H.264/AVC에서 사용자는 필터의 세기를 다소 조절할 수 있다. Clip()함수는 계산하여 얻은 값을 0과 51사이의 값으로 제한하고, 위의 식에서 얻은 indexA, B를 표준안의 표에 대입하여 α와 β를 구한다. QP값이 작을 경우 경계주위 샘플 값의 변화는 실제 이미지의 경계성분으로 인한 것일 가능성이 크므로 α, β의 값이 작아 필터의 적용빈도가 낮아지고, 반대로 QP값이 클 경우 그로 인한 블록왜곡이 일어날 가능성이 크므로 필터를 적용할 가능성이 커지게 된다. 위를 통해 필터링 결정은 QP값에 의해 많이 좌우된다는 사실을 알 수 있다[6].

2. 필터링 수행

가. bS = 4 일 때

bS가 가장 큰 4 일 경우 식 (4)의 조건이 맞으면 식 (5)와 같이 p2, p0에는 서로 다른 5-tap filter를 적용하고, p1에는 4-tap filter를 적용하여 경계주위 3 개의 샘플 값을 필터링한다.

$$|p_2 - p_0| < \beta,\ |p_0 - q_0| < (\alpha \gg 2) + 2 \quad (4)$$

$$p'_0 = (p_2 + 2p_1 + 2p_0 + 2q_0 + q_1 + 4) \gg 3 \quad (5)$$
$$p'_1 = (p_2 + p_1 + p_0 + q_0 + 2) \gg 2$$
$$p'_2 = (2p_3 + 3p_2 + p_1 + p_0 + q_0 + 4) \gg 3$$

식 (4)의 조건이 맞지 않을 경우 식 (6)과 같이 p0에만 3-tap filter를 적용한다.

$$p'_0 = (2p_1 + p_0 + q_1 + 2) \gg 2 \quad (6)$$

p와 q를 바꾸어 위 과정을 동일하게 적용한다.

나. bS = 1, 2, 3 일 때

bS가 1, 2, 3일 경우 최대 경계 주위 2 개의 샘플 값

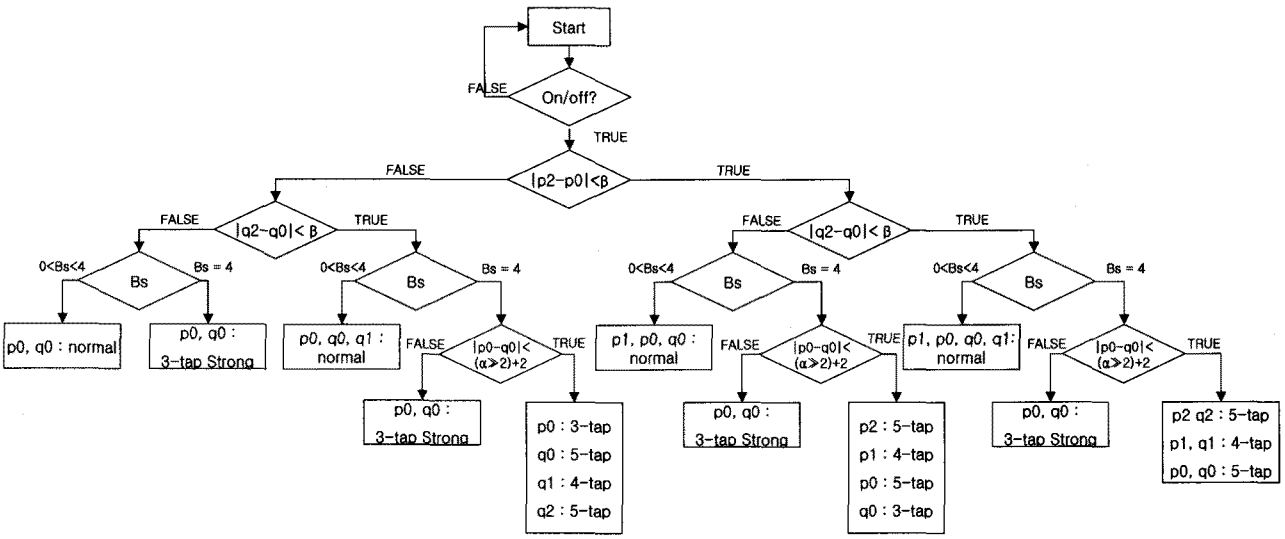

그림 4. 디블록킹 필터의 알고리즘
Fig. 4. Deblocking filter Algorithm.

을 필터링한다. 필터링 과정은 두 부분으로 나뉘는데 먼저 필터링을 위한 기본연산을 한 후, 클리핑 변수 t_{c0}에 의해 그 값을 제한하게 된다.

(1) 기본연산

식 (7-9)은 p0와 q0의 필터링을 위한 기본연산으로, 먼저 Δ_{0i}를 구한 후 클리핑과정을 거쳐 Δ_0를 각 픽셀 값에 더하거나 빼준다.

$$\Delta_{0i} = (4(q_0 - p_0) + (p_1 - q_1) + 4) \gg 3 \quad (7)$$

$$p_0' = p_0 + \Delta_0 \quad (8)$$

$$q_0' = q_0 - \Delta_0 \quad (9)$$

경계주위 샘플 p1과 q1은 각각 식 (10-11)의 조건이 참일 경우 식(12-15)를 통해 필터링된다. p0, q0의 필터링과정과 비슷하게 식(12-13)을 통해 Δ_{p1i}, Δ_{q1i}를 구한 후, 클리핑과정을 거쳐 각 픽셀값에 더해준다.

$$|p_2 - p_0| < \beta \quad (10)$$

$$|q_2 - q_0| < \beta \quad (11)$$

$$\Delta_{p1i} = (p_2 + ((p_0 + q_0 + 1) \gg 1) - 2p_1) \gg 1 \quad (12)$$

$$\Delta_{q1i} = (q_2 + ((p_0 + q_0 + 1) \gg 1) + 2q_1) \gg 1 \quad (13)$$

$$p_1' = p_1 + \Delta_{p1} \quad (14)$$

$$q_1' = q_1 + \Delta_{q1} \quad (15)$$

(2) 클리핑 과정

기본연산을 거친 Δ_{0i}, Δ_{p1i}, Δ_{q1i}를 클리핑 과정 없이 바로 사용하게 되면 과도한 필터링으로 인해 블러링 현상이 발생할 수 있다[7]. 따라서 표준안에서 정의한 클리핑 변수를 통해 이 값을 제한하는 과정이 필요하다. 식 (16-18)은 각각 Δ_{p1i}, Δ_{q1i}, Δ_0의 클리핑과정을 나타낸다. Δ_{0i}를 제한하기 위한 클리핑 변수 t_c는 휘도샘플의 경우, 위 식 (10-11)의 조건이 맞으면 t_{c0}에 1씩 더하고, 색차샘플의 경우 조건없이 1을 더한 값이다.

$$\Delta_{p1} = Min(Max(-t_{c0}, \Delta_{p1i}), t_{c0}) \quad (16)$$

$$\Delta_{q1} = Min(Max(-t_{c0}, \Delta_{q1i}), t_{c0}) \quad (17)$$

$$\Delta_0 = Min(Max(-t_c, \Delta_{0i}), t_c) \quad (18)$$

그림 4는 디블록킹 필터의 전체적인 순서도를 나타낸 것으로, 먼저 필터의 적용여부를 결정한 다음 각각의 조건에 따라 적용될 필터를 보여준다.

III. 제안된 구조

1. 필터링 스캔 순서

디블록킹 필터의 처리량을 향상시키고 동시에 내부 메모리를 최소화하는 효율적인 디블록킹 필터의 설계를 위해 여러 가지 필터링 순서와 구조가 제안되어왔다[5~13]. Cheng 등은 메모리 참조를 줄이기 위해 그림 5와 같은 필터링 순서를 제안하였다[8]. 이 순서는 표준안과 같이 매크로블록 단위로 하나의 수직경계를 필터링하고 블록을 메모리에 저장하는 것이 아니라 블록단위로 바로 옆의 수직경계를 필터링하여 데이터를 재사용하였다. 하지만 이 방법은 수평경계의 필터링을 위해 블록을 저장하였다가 다시 읽어 와야 하는 문제가 있다. 따라서 Sheng등은 수직경계와 수평경계를 구분하지 않고 필터링하는 그림 6과 같은 순서를 제안하였다[9]. 이 순

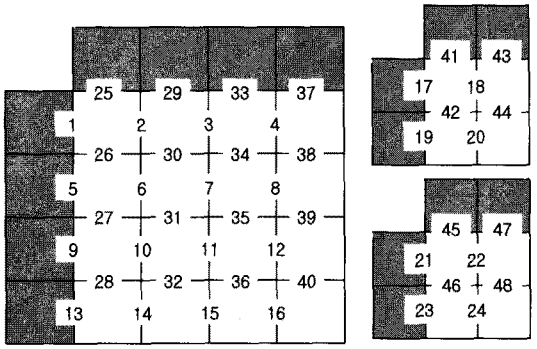

그림 5. [8]에서 제안된 필터링 순서
Fig. 5. Filtering order in reference [8].

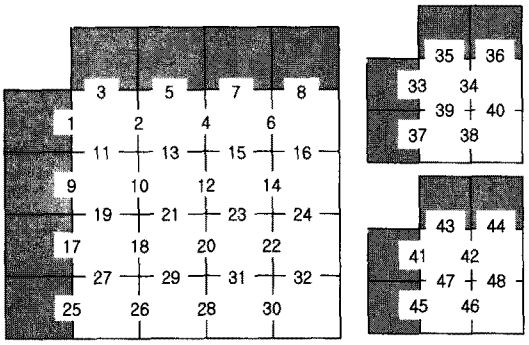

그림 6. [9]에서 제안된 필터링 순서
Fig. 6. Filtering order in reference [9].

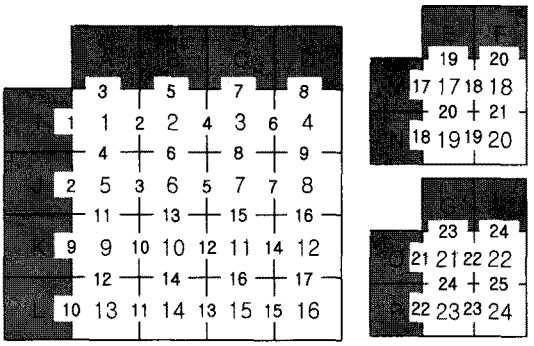

그림 7. 제안된 필터링 순서
Fig. 7. Proposed filtering order.

서는 블록의 양 옆의 수직경계를 필터링한 후, 바로 위 블록과의 수평경계를 필터링함으로써 앞의 방법보다 더 메모리 참조를 줄일 수 있었다.

우리는 위와 같은 방식에서 하나의 필터를 더 사용하여 블록의 위 수평경계와 함께 바로 아래 수평경계까지 가능한한 바로 필터링할 수 있도록 확장한 새로운 필터링 순서를 제안한다. 제안된 필터링 순서는 먼저 두 개의 수직경계를 필터링한 후 블록의 위 수평경계를 필터링하고 마지막 아래 수평경계를 필터링하기 위해서는 아래 블록의 두 수직경계가 필터링되어 있어야 하므로 이를 담당할 필터를 사용하였다. 색차블록의 경우 휘도블록과 동일하게 필터링되므로 여기서는 휘도블록만을 다룬다.

그림 7은 제안된 필터링 순서를 나타낸 것으로, 하나의 블록간의 경계는 4 개의 샘플 경계로 이루어져 4 클럭이 소요된다. 먼저 하나의 필터가 1 번 경계를 필터링 하고난 후 2 번 경계를 필터링할 때부터 두 개의 필터가 동시에 동작하기 시작한다. 두 개의 필터는 각각 서로 독립적이지만 동일한 방법으로 경계를 필터링한다. 하나의 필터는 첫 행과 세 번째 행의 필터링을 수행하고, 다른 하나의 필터는 두 번째 행과 네 번째 행의 필터링을 수행한다. 첫 행의 첫 번째 수직경계가 필터링된 후 두 번째 필터가 같이 동작하게 되는데, 그 이유는 만약 동시에 시작하게 된다면 수직경계의 필터링이 끝난 후, 1 번 블록의 위 수평경계와 아래 수평경계를 동시에 필터링해야 하는 경우가 생기게 되고, 하나의 블록을 두 개의 필터에서 동시에 사용할 수 없기 때문이다. 따라서 두 번째 필터는 첫 번째 필터보다 하나의 블록 경계를 필터링하는 시간만큼 지연을 두어 동작을 시작하게 된다.

2. 제안된 필터 구조

제안된 구조는 필터를 두 개 사용하여 두 개의 샘플 경계를 동시에 필터링하고 모든 데이터는 행으로 4 개의 샘플단위로 32-bit 폭으로 저장되고, 이동한다. 그림 8은 제안된 구조를 나타낸 것으로 f0, f1은 두 개의 필터이고 t0~t5는 수평경계 필터링에 따르는 데이터 재배열을 위한 행열변환(transpose)레지스터이다.

제안된 구조에 사용된 필터는 처리량을 향상시키기 위해 2단 파이프라인으로 구성되었는데 필터의 최장 지연시간을 갖는 임계경로는 기본 연산 과정과 클리핑 과정이 순차적으로 이루어지는 bS가 1, 2, 3일 때 경계주위 4개 픽셀(p1, p0, q0, q1)의 필터링이다. 따라서 첫 클럭에서 기본 연산과 필요한 조건을 연산하는 과정을 먼저 수행하여 내부 레지스터에 저장하고, 다음 클럭에서 조건에 맞는 필터링 연산을 고르고, 나머지 연산을 수행하게 된다. 그림 9는 2단 필터의 내부 구조를 간략히 그린 것이다.

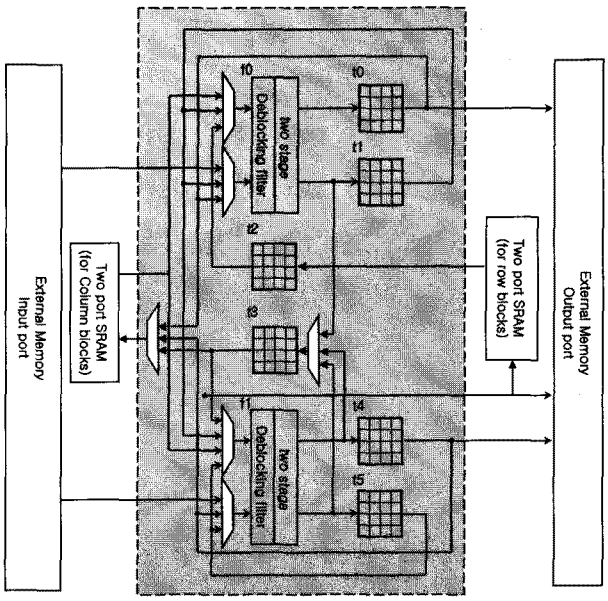

그림 8. 제안된 구조
Fig. 8. Proposed architecture.

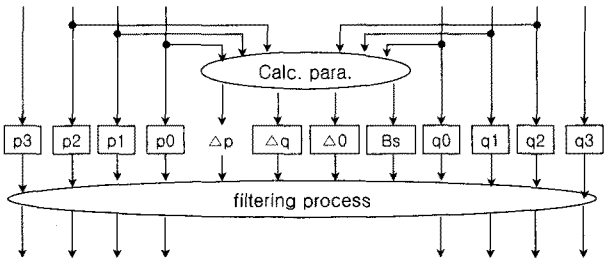

그림 9. 2단 파이프라인 필터
Fig. 9. 2 stage pipelined filter.

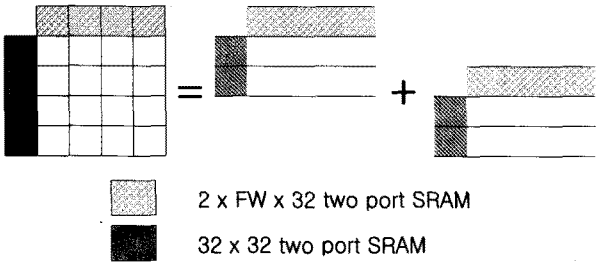

그림 10. SRAM에 저장되는 블록
Fig. 10. Organization of SRAM.

제안된 구조는 두 개의 two-port SRAM을 사용하는데, 각각 매크로블록의 위 블록(그림 7에서 A~H등)을 저장하는 row SRAM과 왼쪽 블록(I~P)을 저장하는 column SRAM이다. 디블록킹 필터는 프레임에서 매크로블록 단위로 필터링을 수행하고, 인접한 위 4 개의 블록과 왼쪽 4 개의 블록을 필요로 한다. 따라서 우리는 프레임 너비만큼의 블록과 왼쪽 4 개의 블록을 SRAM에 저장하여 외부 메모리의 참조를 최소화하였다. 그림 10은 두 개의 SRAM에 저장되는 중간 블록들을 나타낸 것이다. 제안된 구조는 행 방향으로 필터링되므로 매크로블록을 두 부분으로 나누어 생각할 수 있고, row SRAM은 매크로블록의 필터링 중 두 차례에

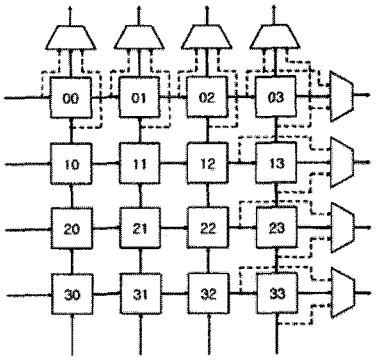

그림 11. 행열변환 레지스터
Fig 11. Transpose register

걸쳐 갱신된다. 위 부분의 필터링을 마치면 5~8번 블록이 저장되고 나머지 아래 부분의 필터링을 마치면 13~16번 블록이 저장된다. column SRAM은 필터링 과정 중 내부에 저장된 블록들이 차례로 4, 8, 12, 16번 블록으로 갱신되어 다음 매크로블록을 필터링할 때 재사용되도록 한다.

그림 11은 제안된 구조에서 사용한 행열변환 레지스터로써 데이터 저장공간의 역할과 수평경계의 필터링 시 필요한 데이터를 재배열하는 역할을 한다. 4 개의 픽셀로 이루어진 하나의 데이터 워드가 4 번 쉬프트하

표 1. 데이터 플로우
Table 1. Data flow.

CLK	t0	t1	t2	t3	t4	t5	SRAM for row block								SRAM for column block							
89-92	G^{-1}	22^{-1}	H^{-1}	21^{-1}	23^{-1}	24^{-1}	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
93-96	H^{-1}	22^{-1}		23^{-1}	21^{-1}	24^{-1}	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1-4	I	1		24^{-1}	22^{-1}		A	B	C	D	E	F	G	H		J	K	L	M	N	O	P
5-8	1	2	A	24^{-1}	J	5		B	C	D	E	F	G	H			K	L	M	N	O	P
9-12	A	2		1	5	6		B	C	D	E	F	G	H			K	L	M	N	O	P
13-16	2	3	B	5	1	6			C	D	E	F	G	H			K	L	M	N	O	P
17-20	B	3		2	6	7	5		C	D	E	F	G	H			K	L	M	N	O	P
21-24	3	4	C	6	2	7	5			D	E	F	G	H			K	L	M	N	O	P
25-28	C	4	D	3	7	8	5	6			E	F	G	H			K	L	M	N	O	P
29-32	D	4		7	3	8	5	6			E	F	G	H			K	L	M	N	O	P
33-36	K	9		8	4		5	6	7		E	F	G	H				L	M	N	O	P
37-40	9	10	5		L	13		6	7	8	E	F	G	H	4				M	N	O	P
41-44	5	10		9	13	14		6	7	8	E	F	G	H	4				M	N	O	P
45-48	10	11	6	13	9	14			7	8	E	F	G	H	4				M	N	O	P
49-52	6	11		10	14	15	13		7	8	E	F	G	H	4				M	N	O	P
53-56	11	12	7	14	10	15	13			8	E	F	G	H	4				M	N	O	P
57-60	7	12	8	11	15	16	13	14			E	F	G	H	4				M	N	O	P
61-64	8	12		15	11	16	13	14			E	F	G	H	4				M	N	O	P
65-68	M	17		16	12		13	14	15		E	F	G	H	4	8				N	O	P
69-72	17	18	E	16	N	19	13	14	15			F	G	H	4	8	12				O	P
73-76	E	18	F	17	19	20	13	14	15				G	H	4	8	12	16			O	P
77-80	F	18		19	17	20	13	14	15				G	H	4	8	12	16			O	P
81-84	O	21		20	18		13	14	15		19		G	H	4	8	12	16				P
85-88	21	22	G	20	P	23	13	14	15		19			H	4	8	12	16	18			
89-92	G	22	H	21	23	24	13	14	15		19				4	8	12	16	18	20		
93-96	H	22		23	21	24	13	14	15		19		23		4	8	12	16	18	20		
1-4	I'	1'		24	22		13	14	15		19		23			8	12	16	18	20		
5-8	1'	2'	A'	24	J'	5'	13	14	15		19		23				12	16	18	20	22	
9-12	A'	2'		1'	5'	6'	13	14	15		19		23				12	16	18	20	22	24

여 입력되면 하나의 4×4블록이 저장되고, 입력한 방향과 수직의 방향으로 데이터를 출력하면 자연스럽게 행열변환의 효과를 얻을 수 있다. 제안된 구조는 파이프라인 필터를 사용하기 때문에 레지스터에 4×4블록이 모두 저장되지 않아도 데이터를 미리 가져와 사용해야 하는 경우가 생긴다. 따라서 이러한 문제를 해결하기 위해 점선 방향으로 데이터를 미리 꺼내어 사용할 수 있도록 설계하였다.

3. 데이터 플로우

제안된 순서에 따라 필터링이 수행되기 위해 다음과 같은 방법으로 데이터가 이동하게 된다. 먼저, 1 번 경계를 필터링하기 위해 column SRAM에서 I 블록을, 외부메모리에서 1 번 블록을 가져와 f0에 입력한다. 필터링이 끝난 I 블록은 외부메모리로 출력되고, 1 번 블록은 오른 쪽 수직경계를 필터링하기 위해 t1에 저장된다. 2 번 경계부터 두 개의 필터가 동시에 동작한다. t1에 저장된 1 번 블록과 외부메모리로부터 가져온 2 번 블록이 f0에 입력되고, column SRAM에 저장되어 있던 J 블록과 외부메모리의 5 번 블록이 f1에 입력된다. 동시에 다음 클록에 3 번 수평경계를 필터링하기 위해 A 블록을 행열변환해 주어야 하므로 row SRAM에서 t2로 가져와 저장해둔다. f0에서 출력된 1 번 블록은 위쪽 수평 경계를 필터링해야 하므로 t0에 저장하여 행열변환할 수 있도록 하고, 2 번 블록은 다음 수직경계를 필터링 하기 위해 t1에 저장한다. f1의 출력, J 블록은 필터링이 끝났으므로 외부메모리에 출력하고, 5 번 블록은 남은 수직경계를 위해 t5에 저장한다. 3 번 수평경계를 위해 A와 1 번 블록을 행열변환 레지스터에서 수직방향으로 꺼내어 f0에 입력하고, 5 번, 6 번 블록을 f1에 입력한다. 모든 경계가 다 필터링된 A 블록은 외부메모리에 다시 저장하기 전에 다시 행열변환해 주어야 하므로 t0에 저장하고, 1 번 블록은 아래 수평경계의 필터링이 남았으므로 f1에 입력하기 위해 t3에 저장한다. f1의 출력, 5 번 블록은 t4에, 6 번 블록은 t5에 저장한다. f1의 필터링 과정은 f0의 과정과 동일하다. t3에 저장된 1 번 블록은 행열변환되어 5 번 블록과 함께 f1에 입력되고, 외부메모리에 출력하기 전에 다시 t4에 저장하여 행열변환한다. 아래 수평경계의 필터링이 남은 5 번 블록은 f0에 의해 아래 9 번 블록의 두 수직경계가 필터링될 때까지 잠시 row SRAM에 저장된다. 같은 방식으로 6~8 번 블록이 row SRAM에 저장되고, 하나의 매크로블록이 필터링된 후에는 13~16 번 블록이 저장된다. 표 1은 매크로블록을 필터링하면서 두 개의 SRAM과 내부 레지스터에 저장되는 블록을 나타낸 것이다. 표에서의 클록은 하나의 매크로블록을 단위로 세었고, " $^{-1}$ "표시는 이전 매크로블록의 블록을 나타내며 " ' "표시는 다음 매크로블록의 블록을 나타낸다. 표에서 볼 수 있듯이 제안된 구조는 앞, 뒤 매크로블록의 필터링 과정과 맞물려 연속적으로 이루어지므로 두 개의 필터가 동작하기까지 4 클럭의 초기지연은 전체 성능에 영향을 미치지 않는다.

프레임 가장자리에서 디블록킹 필터는 가능한 블록만 필터링하고 인접한 블록이 없는 나머지 블록은 필터를 그냥 통과할 뿐 필터링하지 않는 것 외에 동일하게 동작한다. 프레임의 필터링 과정 중 두 개의 SRAM이 채워지고 비워지는 과정은 다음과 같다. 필터링을 시작할 때 SRAM은 초기화되어 있다. column SRAM은 왼쪽 가장자리 매크로블록의 필터링이 끝나면 4 개의 블록이 저장되어 다음 매크로블록을 필터링할 때부터 사용되고, 오른쪽 가장자리 매크로블록을 필터링할 때는 블록을 저장하지 않고 바로 출력하여 SRAM을 비운다. row SRAM은 위 가장자리 매크로블록을 필터링할 때는 인접한 블록이 없으므로 참조될 필요가 없으며, 다

표 2. 기존의 필터 구조와 제안된 필터의 성능비교
Table 2. Performance comparison with the various deblocking filters.

	[6]	[9]	[10]	[11]	[12]	[13]	proposed
# of filter	2	1	1	1	1	1	2
tech.	0.18㎛	0.25㎛	0.25㎛	0.25㎛	0.18㎛	0.13㎛	0.18㎛
gate	18.43K	24K	20.66K	13.41K	20.9K	7.5K	20.9K
frequency	200㎒	100㎒	100㎒	100㎒	100㎒	200㎒	200㎒
clock/MB	240	446	614	300	192	192	96
time/MB	1200ns	4460ns	6140ns	3000ns	1920ns	960ns	480ns
registers	32 × 32	32 × 32	16 × 32	24 × 32	8 × 32	40 × 32	24 × 32

음행을 필터링할 때 필요한 윗 블록들을 저장한다. 아래 가장자리의 매크로블록을 필터링할 때에는 저장하지 않고 바로 출력한다.

IV. 설계 및 성능분석

제안된 구조는 Verilog HDL로 모델링 되었으며 Synopsys Design Compiler를 이용하여 게이트 수준의 합성 결과를 얻었다. 동부아남 0.18㎛ 표준 셀 라이브러리를 사용하여 합성한 결과 최대 동작 주파수는 200㎒이며 총 20.9K 게이트가 사용되었다.

제안된 구조의 필터링 순서에 따르면, 4:2:0 이미지포맷(16x16 휘도블록 하나와 8x8 색차블록 두 개)에서 하나의 매크로블록을 필터링하는데 24x4 = 96 클럭이 소요된다. 프레임의 마지막 매크로블록 필터링은 수평경계 필터링으로 끝나므로 마지막 출력블록을 행열변환하여 저장해야 한다. 따라서 4 클럭이 더 소요된다. 표 2은 다른 논문의 구조와 성능을 비교한 것이다.

[9]와 [10]에서 제안된 구조는 하나의 필터를 사용하면서 총 게이트수가 많고 필터링성능도 떨어진다. [11]에서 제안된 구조는 비교적 게이트를 줄이며 성능도 다소 개선되었지만 여전히 성능이 떨어지며, [12]에서 제안된 구조는 하나의 필터로 비교적 최적화된 성능을 보이지만 게이트수가 많고, [13]에서 제안된 구조는 하나의 필터로 적은 게이트 수로 성능을 개선하였지만 내부의 버퍼를 많이 사용하였다. [6]에서 제안된 구조는 2개의 필터를 사용하여 약간 적은 게이트를 보이지만 2개의 필터가 동시에 동작하기까지 초기화 절차가 필요하여 두 배의 성능을 내지 못하며, 시스템을 변경해야 하는 부담이 있다. 반면 제안된 구조는 2 개의 필터가 동시에 동작하기까지 프레임당 4 클럭만이 소요될 뿐 매크로블록을 필터링할 때마다 초기화과정이 필요하지 않고, 기존의 시스템에 바로 적용이 가능하며, 비교적 적은 내부 버퍼를 사용하고 있다.

V. 결 론

본 논문은 성능과 데이터의 재사용을 최대한으로 하여 외부메모리의 참조횟수를 감소시키는 효율적인 디블록킹 필터의 구조를 제안하고 최소한의 내부 메모리를 갖도록 설계하였다. 제안된 구조는 복잡도가 약간 증가하였지만, 적은 초기화 과정을 거친 후 두 개의 필터가 동시에 동작하여 두 개의 필터로 최적화된 성능을 보인다. 또한 비교적 적은 내부 버퍼를 가지며, 기존 시스템의 변경 없이 적용이 가능하다. 제안된 구조는 동부아남 0.18㎛ 표준 셀 라이브러리를 사용하여 합성한 결과 최대 동작 주파수는 200㎒이며 총 20.9K 게이트가 사용되었다.

감사의 글

저자들은 본 연구를 위하여 설계 환경을 제공하여준 IDEC에 감사드린다.

참 고 문 헌

[1] Joint Video Team(JVT) of ITU-T VCEG and ISO/IEC MPEG, *Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification*, ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, May 2003.

[2] Iain E. G. Richardson, *H.264 and MPEG-4*, WILEY, 2004.

[3] M. Parlak, I. Hamzaoglu, "An Efficient Hardware Architecture for H.264 Adaptive Deblocking FilterAlgorithm", *Proceedings of the first NASA/ESA conference on Adaptive Hardware and Systems*, pp.381-385, June 2006.

[4] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, "H.264/AVC baseline profile decoder complexity analysis", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 13, No. 7, pp. 704-716, July 2003.

[5] T. M. Liu, W. P. Lee, C. Y. Lee, "An In/Post-Loop Deblocking Filter with Hybrid Filtering Schedule," *Circuits and Systems for video Technology*, Vol. 17, No. 7, pp. 937-943, July, 2007.

[6] C. M. Chen and C. H. Chen, "A Memory Efficient Architecture for Deblocking filter in H.264 Using Vertical Processing Order," *Proceedings of the 2005 International Conference on Intelligent Sensors*, pp. 361-366, Dec. 2005.

[7] P. List, A. Joch, J. Lainema, G. Bjontegaard, and M. Karczewicz, "Adaptive Deblocking Filter," *IEEE Transactions On Circuits and Systems for video Technology*, Vol. 13, No. 7, pp. 614-619, July, 2003.

[8] C. C. Cheng and T. S. Chang, "A Hardware Efficient Deblocking Filters for H.264/AVC," *IEEE Consumer Electronics, 2005 Digest of Technical Papers*, pp.235-236, Jan. 2005.
[9] B. Sheng, W. Gao and D. Wu, "An Implemented Architecture of Deblocking Filter for H.264/AVC," *IEEE International Conference on Image Processing*, pp. 665-668, 2004.
[10] Y. Huang, T. Chen, B. Hsieh, Y. Wang, T. Chang, and L. Chen, "Architecture Design for Deblocking Filter in H.264/JVT/AVC," *International conference on Multimedia and Expo*, Vol. 1, pp. I-693-6, July, 2003.
[11] C. C. Cheng and T. S. Chang, "An In-Place Architecture for the Deblocking Filter in H.264/AVC," *IEEE Transactions on Circuits and Systems-II, Express Briefs*, Vol. 53, No. 7, pp. 530-534, July 2006.
[12] S. Y. Shih, C. R. Chang and Y. L, Lin, "A Near Optimal Deblocking Filter for H.264 Advanced Video Codiing," *IEEE Asia and South Pacific Conference on Design Automation*, pp. 170-175 Jan. 2006.
[13] G. Khurana and A. A. Kassim, "A Pipelined Hardware Implementation of In-loop Deblocking Filter in H.264/AVC," *IEEE Transactions On Consumer Electronics*, Vol. 52, No. 2, pp. 536-540, May 2006.

저 자 소 개

이 성 만(학생회원)
2008년 가톨릭대학교 정보통신 전자공학과 졸업.
2008년~현재 가톨릭대학교 정보통신전자공학과 석사과정.
<주관심분야 : VLSI 설계, 영상처리>

박 태 근(정회원)-교신저자
1985년 연세대학교 전자공학과 졸업.
1988년 Syracuse Univ. Computer 공학석사 졸업.
1993년 Syracuse Univ. Computer 공학박사 졸업.
1991년~1993년 Coherent Research Inc. USA VLSI 엔지니어
1994년~1998년 현대전자 System IC 연구소 책임연구원
1998년~현재 가톨릭대학교 정보통신전자공학부 정교수
<주관심분야 : VLSI 설계, CAD, 컴퓨터구조>