H.264/AVC의 프레임내 예측 부호화를 위한 부가적인 하드웨어 구조

이수진° 김정길 김명서 김신덕
연세대학교 컴퓨터과학과
{haegy°, tetons, museflute, sdkim}@yonsei.ac.kr

An Additional Hardware Architecture for H.264/AVC Intra-Prediction

Sujin Lee° Cheongghil Kim Myoungseo Kim, Shindug Kim
Department of Computer Science, Yonsei University

요 약

H.264/AVC의 프레임내 예측기법은 현 매크로블록의 이웃픽셀들로부터 예측값을 추출함으로써 인트라 픽처의 압축률을 높이는데 크게 기여했다. 그러나 모든 매크로블록에 대해 총 17가지의 후보 모드를 검사해야 하기 때문에, 전체 부호화기의 복잡도를 상당히 상승시키는 요인이기도 하다. 본 논문에서는 이 문제를 해결하기 위해, 기존의 움직임 추정 전용 하드웨어로 주로 사용되는 1차원 시스톨릭 어레이 구조에 부가적인 하드웨어를 장착하여, 움직임 추정뿐만 아니라 프레임 내 예측까지 가능한 하드웨어 구조를 제안한다. 병렬적으로 끊김이 없는 수행을 위해 프레임내 예측 알고리즘을 약간 수정했으나, 이것은 화질이나 비트스트림 크기에 영향을 거의 미치지 않는다. 제안된 구조는 연산에 대한 명령어 개수로 비교할 때, ARM 기반 시스템에서 얻을 수 있는 성능의 10배에서 40배에 달하는 높은 성능을 보여준다.

1. 서 론

최신의 동영상 압축 표준인 H.264/AVC는 이전의 표준안들에 비해 월등히 뛰어난 압축효율과 성능을 보이는 기법들을 많이 채택했다. 그 기법들 중 하나인 프레임내 예측기법은 특히 인트라 픽처의 압축률을 크게 높이는데 공헌했다. 예측을 위한 모드는 4x4 휘도블록에 대해 9개, 16x16 휘도블록에 대해 4개, 8x8 색도블록에 대해 4개의 후보모드로 총 17개인데, JM (Joint Model)[1]의 참고 소프트웨어에 의하면 프레임내 예측의 후보모드 검사는 모든 매크로블록마다 이루어지므로, 프레임간 예측인 움직임 추정이 그렇듯 상당히 많은 연산량을 필요로 한다.

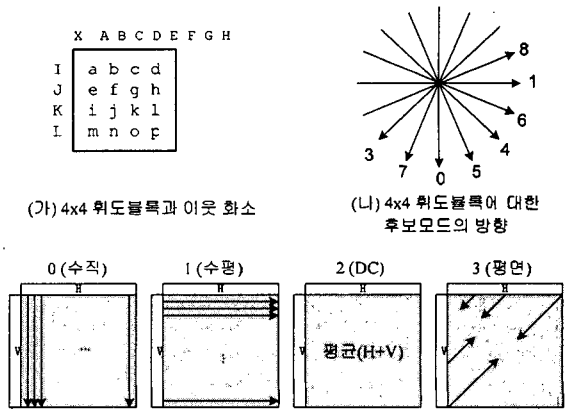
프레임내 예측의 이러한 복잡도를 줄이기 위해, 주로 예측 알고리즘을 개선하는 방식의 연구가 소개되어왔다. [2]에서는 서브샘플링과 한계값을 이용해 연산량을 감소시켰고, [3]의 연구에서는 움직임 추정과 프레임내 예측을 함께 묶어 모드 결정방법을 최적화시키는 방법을 사용했다. [4]에서는 하드웨어를 고려한 프레임내 예측기법의 수정을 제안했고, [5]에서는 경계선의 방향성을 미리 추정해 후보모드의 수를 줄이는 방법을 제시했다. 그리고 [6]에서는 방향성이 고려된 DC예측과 단순화된 후보모드를 이용해 수행할 연산량을 줄였다.

프레임내 예측은 프레임간 예측과 같은 단계에서 진행되고 비슷한 연산을 처리하기 때문에, 프레임간 예측을 위한 전용 하드웨어가 있는 경우 이를 이용해서 처리할 수 있다. 본 논문에서는 프레임내 예측을 위한 별도의 하드웨어 모듈을 구축할 필요 없이, 프레임간 예측용의 전용 1차원 시스톨릭 어레이 구조에 몇 개의 멀티플렉서와 덧셈기를 추가하여 프레임내 예측을 수행하는 하드웨어 구조를 제안한다. 본 구조를 이용하면 기존의 시스톨릭 어레이로 프레임간 예측의 복잡도를 줄이는 것에 더해, 프레임내 예측까지 하드웨어로 처리함으로써 전체적인 부호화기의 복잡도를 크게 감소시킬 수 있다. 이를 구현하기 위해 알고리즘 부분에서 약간의 수정이 이루어지나, 영상의 화질과 크기 면에서 거의 영향을 끼치지 않는다.

2. 프레임내 예측 알고리즘

2.1 H.264/AVC 프레임내 예측 알고리즘

H.264/AVC의 프레임내 예측 알고리즘은 4x4 휘도블록과 16x16 휘도블록, 8x8 색도블록에 각각 다른 후보모드를 제시하고 있다. 그림 1은 각 블록에 따른 후보모드의 종류를 나타낸다.



(가) 4x4 휘도블록과 이웃 화소

(나) 4x4 휘도블록에 대한 후보모드의 방향

(다) 16x16 휘도블록과 8x8 색도블록의 후보 모드

그림 1. 프레임내 예측에서 사용하는 후보모드

4x4 휘도블록의 경우 9가지의 후보모드는 (가)에 보이는 A부터 L까지의 이웃화소로부터 보간법의 방법으로 계산되며, (나)가 8가지 모드의 보간되는 방향을 나타내고 있다. (나)에 없는 2번째 모드(DC 모드)는 이웃화소들의 평균값으로 구성된다. 다음은 사용하는 이웃화소의 개수에 따른 보간법의 연산을 보여준다.

1) (I+2J+K)/4 : 세 개의 이웃화소를 사용할 때.
2) (I+J)/2 : 두 개의 이웃화소를 사용할 때.
3) I : 하나의 이웃화소를 사용할 때.

(다)는 16x16 휘도블록과 8x8 색도블록에 대한 4가지의 후보모드를 보여준다. 이중 3번째 후보모드인 평면모드는 17개에서 33개 화소의 복잡한 보간법에 의해 계산된다.

2.2 하드웨어 구현을 위한 프레임내 예측 알고리즘

알고리즘의 수정은 다음 두 가지 방법으로 이루어진다. 첫번째로 2.1장에 소개된 보간법의 경우 정확한 평균을 내기 위해 반올림을 사용하는데, 하드웨어상의 나눗셈은 버림을 의미하므로 반올림 결과값을 내기 위해 1/2을 더하는 과정이 필요하게 된다. 이것은 입력 핀 수를 증가시키거나 클럭 사이클을 추가로 사용하게 하므로, 우리는 하드웨어 구현의 단순성을 위해 이 과정을 생략하고 대신 버림을 사용했다. 두번째로 다른 후보모드에 비해 상당히 많은 계산량을 요구하는 평면모드의 검사를 생략한다. 이것은 하드웨어로 구현될 때 다른 후보모드를 계산하는 것보다 평면모드의 연산 시간이 훨씬 커 병렬성을 유지할 수 없기 때문이다. 이 두 가지 수정은 부호화된 영상의 PSNR(the Peak Signal to Noise Ratio)과 최종 비트스트림 크기에 큰 영향을 미치지 않는다.

표 1. 각 영상의 조건별 PSNR과 비트스트림의 크기

영상	Round+ Plane	PSNR (Y)	PSNR (Cb)	PSNR (Cr)	Total Bytes
Foreman	Y	36.820	39.704	41.787	780824
	N	36.816	39.701	41.786	780412
News	Y	37.786	40.556	41.076	841051
	N	37.791	40.554	41.075	841595
Highway	Y	38.629	37.935	38.794	427136
	N	38.623	37.934	38.793	428333
Grandma	Y	37.242	40.058	40.419	664521
	N	37.250	40.057	40.419	665342

표 1은 반올림과 세번째 모드인 평면모드를 사용할 때와 하지 않을 때 각각의 PSNR과 크기를 나타내고 있다. H.264/AVC의 참고 소프트웨어인 JM 8.2를 수정해, QCIF크기의 영상인 Foreman, News, Highway, Grandma를 입력으로 사용했다. 프레임내 예측 알고리즘의 수정이 미치는 영향을 최대한 보여주기 위해 전부 프레임내 예측만 사용했고, 프레임당 30초의 속도와 28의 양자화 계수로 250프레임까지 압축했다.

3. 프레임내 예측을 위한 하드웨어 구조

본 논문에서 제안하는 프레임내 예측을 위한 하드웨어구조의 핵심은 기존의 프레임간 예측을 위한 1차원 시스톨릭 어레이를 그대로 이용한다는 것이다. 프레임내 예측도 SAD(Sum of Absolute Differences)의 계산을 필요로 하기 때문에, 입출력 제어와 추가 계산 등을 위한 부가적인 구조만 추가하면 쉽게 AD 연산을 위한 어레이의 Processing Elements(PEs)를 재이용할 수 있다.

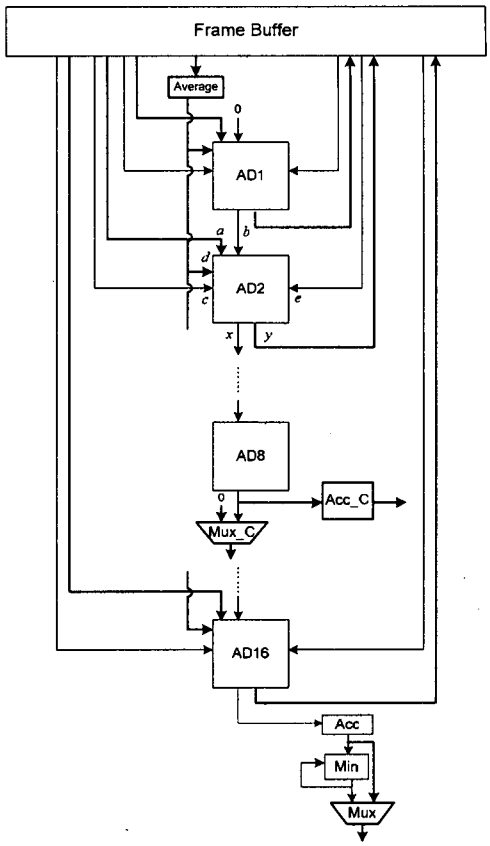


그림 2. 프레임내 예측을 위해 변형된 어레이 구조

프레임내 예측은 두가지 과정이 필요하다. 첫번째 과정은 후보모드에 따른 예측화소(predictor)를 만들어내는 것이고, 두번째는 예측화소와 현재 매크로블록간의 SAD연산을 수행하는 것이다. 예측화소는 이웃화소가 그대로 쓰이는 경우도 있지만 (0번째와 1번째 후보모드), 대부분은 보간법의 계산이 필요하다. 이 보간계산을 위해 PE 내부에 입출력을 제어하는 멀티플렉서의 추가가 필요하며, 자세한 내용은 그림 3 (가)에 나와있다. 그림 2와 3에서 두꺼운 선으로 그려진 부분이 기존의 어레이에 추가된 부분이다. 예측화소를 보간법에 의해 계산할 때는, a와 c, e의 입력을 사용해 이웃픽셀이 2개인지 3개인지에 따라 해당하는 와이어시프트 나눗셈을 거쳐 y로 결과를 출력한다. 또 DC모드에서 요구하는 이웃화소들의 평균값 계산을 위해, 그림 2와 3 (나)의 Average 유닛을 두어 다른 후보모드를 계산하는 동안 화소의 합을 누적시켜 평균을 구한다. 이 때는 c 대신 d의 입력을 이용한다. SAD 연산을 할 때는 c와 e를 사용해 AD를 계산한 후 부분합 입력인 b와 더해 x로 출력한다.

표 2는 4x4 휘도블록의 프레임내 예측을 위한 각 PE 별 입력 화소를 매 사이클마다 나타내고 있다. 어두운 부분의 A부터 L은 그림 1 (가)의 이웃화소이며, 이것은 예측화소를 구하기 위한 입력이다. M의 표시는 M0_1의 경우 0번째 모드의 1번째 화소에 대한 AD 연산을 의미하는 것이다. 16x16 휘도블록과 8x8 색도블록도 유사한 방식으로 진행된다.

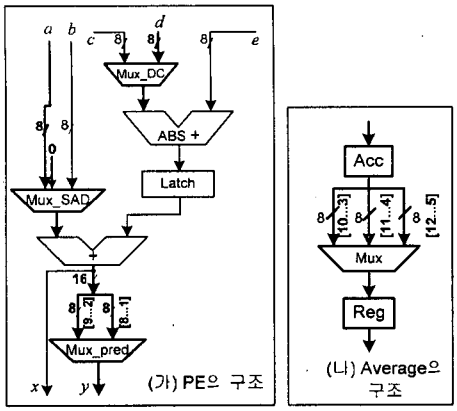


그림 3. PE와 Average 유닛의 구조

표 2. 4x4 휘도블록 프레임내 예측의 입력 데이터 흐름

Clk	t+1	t+2	t+3	t+4	t+5	t+6	t+7	t+8
AD1	ACB	IAX	XA0	IX0	AB0	JI0	M0_1	M1_1
AD2	M8_2	BDC	XBA	AB0	IAX	BC0	KIJ	M0_2
AD3	M7_3	M8_3	CED	ACB	BC0	XBA	CD0	KJ0
AD4	M6_4	M7_4	M8_4	DFE	BDC	CD0	ACB	DE0
AD5	M5_5	M6_5	M7_5	M8_5	BDC	JXI	IAX	JI0
AD6	M4_6	M5_6	M6_6	M7_6	M8_6	CED	IAX	XBA
...	...	...	...	...	...	...	...	...
AD14	IAX	LJK	CED	L00	M0_14	M1_14	M2_14	M3_14
AD15	JXI	XBA	KJ0	DFE	L00	M0_15	M1_15	M2_15
AD16	GHH	IAX	ACB	KIJ	EGF	L00	M0_16	M1_16
Acc	A	B	C	D	I	J	K	L

4. 성능 분석

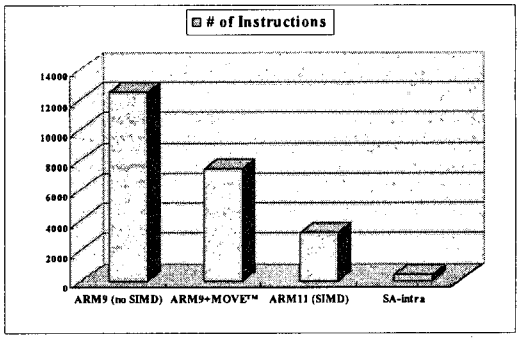
다음의 수식 (1)은 한 매크로블록당 프레임내 예측을 수행하기 위해 걸리는 시간을 표시한다.

$$T_{MB} = 16(P_{4\times4} + SAD_{4\times4}) + (P_{16\times16} + SAD_{16\times16}) + 2(P_{8\times8} + SAD_{8\times8}). \quad (1)$$

본 연구의 하드웨어 구조의 성능을 평가하기 위해, 우리는 SIMD 명령어를 가지고 있는 ARM11이 프레임내 예측을 수행했을 때의 명령어 개수와 비교했다. ARM11을 선택한 것은, ARM11의 SIMD 명령어 집합에 USAD8과 USADA8이란 명령어가 있어서, 특별한 하드웨어의 지원이 없어도 4개 화소의 AD 계산을 한번에 할 수 있는 효율적인 구조를 내장하고 있기 때문이다. 따라서 우리의 구조를 적용시켰을 때와 아닐 때 각각 필요로 하는 명령어의 수를 비교함으로써, 간접적으로 수행시간을 비교할 수 있다. 그림 4의 (가)는 ARM의 어셈블리어를 이용해 구현한 프레임내 예측 알고리즘의 일부이며, (나)는 각 하드웨어를 적용시켰을 경우 명령어의 개수를 비교한 그래프이다. 공정한 비교를 위해 SIMD 명령어가 없는 ARM9과 ARM에서 개발된 움직임 추정용(SAD 연산용) 코프로세서인 MOVE™를 연결했을 때 추정되는 명령어 개수도 함께 비교했다. SA-intra가 본 연구에서 제시한 하드웨어 구조이며, ARM의 SIMD 연산이나 MOVE™보다 프레임내 예측을 10배에서 40배까지 효율적으로 수행하는 것을 볼 수 있다.

```
ARM11 Code Example. Mode 7 of 4x4 Luma Intra-Prediction
; Predictor-calculation
LDRB    r2, [r0]              ; Load a pixel A
LDRB    r3, [r0,#1]           ; Load B
...
ADD     r10, r3, r4           ; B+C.
MOV     r12, r10, LSR #1      ; (B+C)/2.
STRB    r12, [r1, #1]         ; predictor-array7[0][1].
STRB    r12, [r1, #8]         ; predictor-array7[2][0].
...
; SAD operation
LDMIA   r1!, {r4-r7}  ; Load 16 predictors in four register.
LDMIA   r2!, {r8-r11} ; Load 16 current pixels in four register.
USAD8   r12, r8, r4           ; AD between the first rows.
USADA8  r12, r9, r5, r12      ; the second rows.
...
```

(가) ARM 어셈블러 코드



(나) 프레임내 예측의 명령어 개수의 비교

그림 4. ARM 어셈블리 코드와 성능 평가

5. 결론

본 연구에서 제안한 프레임내 예측을 위한 부가적인 하드웨어 구조는, 프레임간 예측을 위해 필요한 어레이구조를 간단한 구조의 추가로 재사용할 수 있다는 것이 가장 큰 특징이다. 즉 H.264/AVC 부호화를 위한 프레임내 예측의 수행을 적은 비용의 하드웨어를 추가하는 것에 의해, 추가하지 않은 ARM기반의 하드웨어보다 명령어 개수에 있어10배에서 40배까지 빠르게 수행할 수 있다.

참고문헌

[1] Joint Video Team (JVT), " Reference Software JM8.2," 2004.
[2] B. Meng, O. C. Au, C. W. Wong, and H. K. Lam, " Efficient Intra-Prediction Algorithm in H.264," Proc. of Int' l Conf. on Image Processing, Vol. 3, pp. 837-840, 2003.
[3] P. Yin, H. Tourapis, A. Tourapis, and J. Boyce, " Fast Mode Decision and Motion Estimation for JVT/H.264," Proc. of Int' l Conf. on Image Processing, Vol. 3, pp. 853-856, 2003.
[4] T.C. Wang, Y.W. Huang, H.C. Fang, and L.G. Chen, L.G, " Performance Analysis of Hardware Oriented Algorithm Modifications in H.264," Proc. of IEEE Int' l Conf. on Multimedia and Expo, Vol. 3, pp. 601-604, 2003.
[5] F. Pan, X. Lin, S. Rahardja, K.P. Lim, and Z.G. Li, " A Directional Field Based Fast Intra Mode Decision Algorithm for H.264 Video Coding," Proc. of IEEE Int' l Conf. on Multimedia and Expo, Vol. 2, pp. 1147-1150, 2004.
[6] Z. Nan, Y. Baocai, K. Dehui, and Y. Wenying, " Spatial Prediction Based Intra-Coding," Proc. of IEEE Int' l Conf. on Multimedia and Expo, Vol. 1, pp. 97-100, 2004.