논문 2009-46SD-7-8

H.264/AVC 동영상 코덱용 고성능 움직임 추정 회로 설계

(Design of High-Performance Motion Estimation Circuit for H.264/AVC Video CODEC)

이 선 영*, 조 경 순**

(Seonyoung Lee and Kyeongsoon Cho)

요 약

H.264/AVC 코덱에 사용되는 움직임 추정은 다중 참조 프레임과 다양한 가변 블록을 이용하기 때문에 복잡하고 많은 연산을 필요로 한다. 본 논문에서는 이러한 문제를 해결하기 위해 다중 참조 프레임 선택, 블록 매칭, 블록 모드 결정, 움직임 벡터 예측을 고속으로 처리하는 방법을 바탕으로 동작 속도가 빠른 정수 화소 움직임 추정 회로 구조를 제안한다. 또한 부화소 움직임 추정을 위한 고성능 보간 회로 구조도 제안한다. 제안한 회로는 Verilog HDL을 이용하여 RTL로 기술하였고, 130nm 표준 셀 라이브러리를 이용하여 합성하였다. 정수 화소 움직임 추정 회로는 77,600 게이트와 4개의 32x8x32-비트 듀얼-포트 SRAM으로 구현되었고 최대 동작 주파수는 161MHz이며 D1(720x480)급 칼라 영상을 1초에 51장 까지 처리할 수 있다. 부화소 움직임 추정 회로는 22,478 게이트로 구현되었고 최대 동작주파수 200MHz에서 1080HD(1,920x1,088)급 칼라 영상을 1초에 69장 까지 처리할 수 있다.

Abstract

Motion estimation for H.264/AVC video CODEC is very complex and requires a huge amount of computational efforts because it uses multiple reference frames and variable block sizes. We propose the architecture of high-performance integer-pixel motion estimation circuit based on fast algorithms for multiple reference frame selection, block matching, block mode decision and motion vector estimation. We also propose the architecture of high-performance interpolation circuit for sub-pixel motion estimation. We described the RTL circuit in Verilog HDL and synthesized the gate-level circuit using 130nm standard cell library. The integer-pixel motion estimation circuit consists of 77,600 logic gates and four 32x8x32-bit dual-port SRAM's. It has the maximum operating frequency of 161MHz and can process up to 51 D1 (720x480) color image frames per second. The fractional motion estimation circuit consists of 22,478 logic gates. It has the maximum operating frequency of 200MHz and can process up to 69 1080HD (1,920x1,088) color image frames per second.

Keywords: H.264/AVC, 움직임 추정, 참조 프레임 선택, 블록 매칭, 블록 모드 결정, 보간

I. 서 론

동영상 압축 표준들은 동영상 데이터에 존재하는 중복성을 제거하여 데이터의 압축률을 향상시킨다. 동일 프레임 내에서 화소들 간의 통계적 발생 확률에 의한 통계적 중복성과 공간적 중복성, 그리고 프레임 사이에 존재하는 시간적 중복성은 동영상 데이터의 대표적인 특징들이다. 이 중에서 동영상의 압축률을 가장 효과적으로 높일 수 있는 시간적 중복성 제거는 이전 영상과 현재 영상의 유사성을 이용하여 움직임 추정 및 보상을 통해 이루어진다. H.264/AVC[1]의 움직임 추정은 다중 참조 프레임, 다양한 블록 크기, 1/2, 1/4 단위의 부화소를 적용하여 영상 데이터 압축의 효율을 높인다. 그러나 추가된 방법들로 인해 외부 메모리로부터의 접근 횟수와 연산의 복잡도가 크게 증가하게 된다. 본 논문에

* 학생회원, ** 평생회원, 한국외국어대학교 전자정보공학부
(Department of Electronics and Information Engineering, Hankuk University of Foreign Studies)

※ 이 논문은 2008년 정부(교육인적자원부)의 재원으로 학술진흥재단의 지원을 받아 수행된 연구 (KRF-2008-521-D00321)이고 2009학년도 한국외국어대학교 교내학술연구비의 지원에 의하여 이루어진 것임.
접수일자: 2009년3월19일, 수정완료일: 2009년6월25일

서는 이 문제를 해결하기 위해 고속 탐색의 한 종류인 NTSS[2](New Three Step Search) 방법과 다중 참조 프레임 선택 방법, 다양한 블록 크기 결정 방법, 움직임 벡터 추정 방법을 고속으로 처리할 수 있는 알고리즘을 바탕으로 회로 구조를 제안한다. 또한 부화소 움직임 추정에서 가장 많은 연산량을 차지하는 부화소 보간을 위해 고성능 보간 회로 구조도 제안한다. 제안된 회로 구조의 동작을 확인하기 위해 H.264/AVC 코덱용 움직임 추정 회로를 RTL(Register Transfer Level)로 구현하고 논리 합성을 통하여 성능을 확인하였다.

본 논문은 4개의 절로 구성되어 있다. Ⅱ절에서는 본 논문에서 사용되는 알고리즘을 선택한 배경에 대해 설명하고 Ⅲ절에서는 움직임 추정 회로의 구조를 제안한다. Ⅳ절에서는 제안한 구조의 성능 분석과 실험 결과를 기술하고 있으며, Ⅴ절에서는 결론을 제시한다.

Ⅱ. 고속 움직임 추정 알고리즘

1. 다중 참조 프레임 선택

이전의 동영상 압축 표준에서는 시간적 중복성을 제거하기 위해 한 장의 참조 프레임을 탐색하여 움직임을 추정한다. 반면 H.264/AVC의 움직임 추정은 여러 장의 참조 프레임에 대해 탐색을 수행한다. 이러한 다중 참조 프레임 탐색은 참조하는 각 프레임의 가변 블록 크기(4x4, 8x4, 4x8, 8x8, 16x8, 8x16, 16x16) 마다 탐색하여 최적의 매칭 블록이 있는 프레임을 선택한다. 이 과정은 메모리 접근과 움직임 추정 연산의 복잡도를 크게 증가시키는 원인이 된다.

본 논문에서는 여러 개의 참조 프레임에 존재하는 모든 가용 블록에 대해 탐색을 하지 않고 움직임 벡터의 확률적인 분포를 이용하는 고속 참조 프레임 선택 알고리즘을 사용하였다. 이 알고리즘은 그림 1과 같이 CBP(Center-Biased Pattern), SSP(Small Square Pattern), LSP(Large Square Pattern), LDP(Large Diamond Pattern) 등과 같은 참조 프레임 선택 패턴을

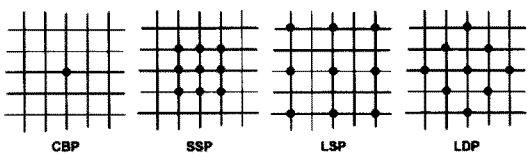

그림 1. 참조 프레임 선택을 위한 예측 패턴들
Fig. 1. Predicted patterns for reference frame selection.

표 1. 참조 프레임 선택 패턴의 비교
Table 1. Comparison of reference frame selection patterns.

	참조 프레임 선택 패턴			
	CBP	LSP	LDP	SSP
SAD 저하	0.028	0.040	0.026	0.006
복잡도 감소 (%)	79.64	76.8	76.8	76.8

이용한다. 표 1[3]은 모든 탐색점을 이용하는 경우와 비교했을 때 SAD(Sum of Absolute Differences)와 연산 복잡도가 얼마나 차이가 나는지 나타낸 것이다. 이 결과를 토대로 SAD 측면에서 손해가 가장 적고 본 논문에서 이용하고자 하는 고속 탐색 방법인 NTSS에 쉽게 적용할 수 있는 SSP를 다중 참조 프레임 탐색 방법으로 선택하였다.

2. 블록 매칭

움직임 추정에서 블록 매칭은 동영상 데이터 압축에서 가장 중요한 역할을 한다. 가장 좋은 압축 성능을 보이는 블록 매칭 방법은 FS(Full Search)이다. 그러나 이 방법은 너무 많은 연산량을 필요로 하기 때문에 이를 극복하기 위한 고속 블록 매칭 방법들이 제안되었다. 그 중에서 하드웨어 구현에 많이 사용되는 고속 블록 매칭 방법은 TSS(Three Step Search)와 NTSS 이다. 표 2[2]와 같이 NTSS는 다른 탐색 알고리즘과 비교하여 FS와 가장 근접한 성능을 보이고 있다. 다른 고속 탐색 알고리즘들은 탐색영역이 좁은 경우 좋은 성능을 보이지만 NTSS는 탐색영역이 좁거나 넓은 경우에 모두 우수한 성능을 보인다.

NTSS의 탐색점은 그림 2와 같다. 초기 단계에서는 탐색영역 중심의 9개 점(1~9)과 외각의 8개 점(1-1~4-1, 6-1~9-1), 총 17개 점에 대해 탐색을 수행한다. 만약 SAD 값의 최소가 되는 지점이 현재 탐색영역의 중심(예를 들어 탐색점 5)이 되면 탐색을 종료한다. 만약 가장 작은 SAD 값이 1~4 또는 6~9의 점에서 발생

표 2. 블록 매칭 알고리즘 비교
Table 2. Comparison of block matching algorithms.

	블록 매칭 방법			
	FS	TSS	Hexagon Search	NTSS
평균 PSNR(dB)	32.32	32.30	32.28	32.32
평균 SAD	1011.36	1013.43	1016.90	1011.18

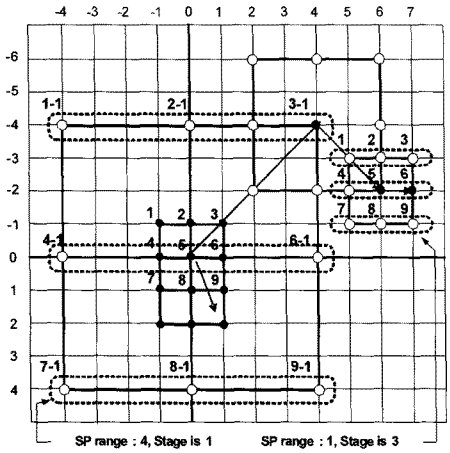

그림 2. NTSS의 탐색점
Fig. 2. Search points of NTSS.

하면 그 점을 중심으로 주변 8개 점에 대해 탐색을 다시 수행하고 탐색을 종료한다. 만약 SAD가 최소인 점이 외곽의 1-1~4-1 또는 6-1~9-1인 경우, 최소가 되는 점을 중심으로 탐색영역을 절반으로 줄여 8개 점에 대해 다시 탐색을 수행하고 앞의 탐색 과정을 수행한다. 이러한 과정은 SAD 최소점이 탐색영역의 중심이 되거나 탐색영역이 1이 될 때까지 반복한다.

3. 블록 모드 결정과 움직임 벡터 추정

H.264/AVC는 7가지 가변 블록 크기(4x4, 8x4, 4x8, 8x8, 16x8, 8x16, 16x16)를 지원한다. 다양한 가변 블록

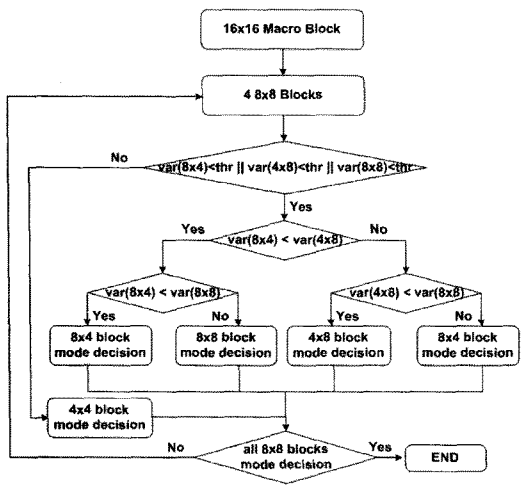

그림 3. 블록 모드 결정 순서도
Fig. 3. Flow chart of block mode decision.

크기를 지원하기 위해서는 계산의 복잡도가 크게 증가하기 때문에 고속 블록 모드 결정 알고리즘이 요구된다. 대부분의 고속 블록 모드 결정 알고리즘들은 움직임 벡터의 상관관계를 이용한다. 본 논문에서는 4x4 블록 단위의 16개 움직임 벡터들(MV_{01}~MV_{16})의 분산을 이용하는 알고리즘[4]을 선택하였다. 먼저 각 4x4 블록으로부터 움직임 벡터 16개를 구하고 식 (1)의 분산 값들을 계산한다. 최적의 블록 모드는 식 (1)의 분산 값과 그림 3의 방법에 따라 결정된다. 먼저 4개의 8x8 블록에 대해 16개 4x4 블록의 분산 값들을 이용하여 4x4, 4x8, 8x4, 8x8의 블록 모드를 결정한다. 각 8x8 블록이 모두 8x8 블록으로 결정되면 16x16, 16x8, 8x16 모드에 대한 동일한 연산을 다시 수행한다. 결정된 블록 모드의 움직임 벡터는 MV_{01}~MV_{16}의 평균값 또는 중간값들을 이용하여 계산한다.

$$
\begin{aligned}
var_{12} &= variance(MV_{01}, MV_{02}) \\
var_{34} &= variance(MV_{03}, MV_{04}) \\
var_{13} &= variance(MV_{01}, MV_{03}) \\
var_{24} &= variance(MV_{02}, MV_{04}) \\
var(8x8) &= variance(MV_{01}, MV_{02}, MV_{03}, MV_{04}) \quad (1) \\
var(8x4) &= var_{12} + var_{34} \\
var(4x8) &= var_{13} + var_{24} \\
var(16x16) &= variance(MV_{01}, \cdots, MV_{16})
\end{aligned}
$$

III. 제안한 회로구조

그림 4는 본 논문에서 제안한 움직임 추정의 전체 회로 구조이다. 회로의 입력으로 현재 프레임 데이터와 고속 다중 참조 프레임 선택에서 결정된 이전 참조 프레임의 데이터가 입력된다. 선택된 참조 프레임 데이터는 그림 5의 'Frame Selection' 단계에서 4개의 32x8x32-비트 듀얼-포트 SRAM('SRAM1'~'SRAM4')에 저장된다. SRAM에 저장되어 있던 데이터는 'Input Controller'에 의해 NTSS 연산을 위해 'PE_Array'에 보내지고 여기에서 SAD 값이 계산된다. 계산된 SAD 값들은 'Comp' 회로를 통해 최소 SAD를 갖는 점을 찾아내는데 사용된다. 참조 프레임 선택과 NTSS 연산을 위해 필요한 데이터들은 회로 내부 버퍼인 'Shared Buffer'에 저장된다. 'Main Control'은 회로 전체의 동작을 제어하여 NTSS 연산을 계속 수행할지 아니면 탐색

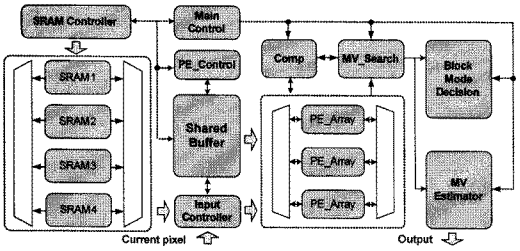

그림 4. 움직임 추정 회로의 전체 구조
Fig. 4. Overall architecture of motion estimation circuit.

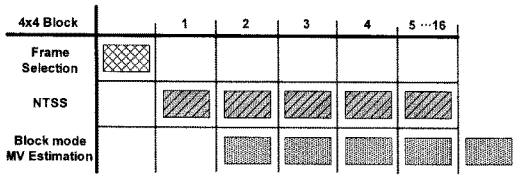

그림 5. 움직임 추정의 파이프라인 단계
Fig. 5. Pipeline stages of motion estimation.

을 종료할지를 결정한다. 탐색이 종료되어 SAD의 최소점이 선택되면, NTSS를 거치면서 얻은 위치 정보를 바탕으로 'MV_search'에서 4x4 블록의 최종 움직임 벡터 값을 결정한다. 최종 움직임 벡터는 그림 5와 같이 NTSS 연산과 블록 모드 결정, 움직임 벡터 추정 연산의 병렬 처리를 통해 얻어진다.

1. New Three Step Search

NTSS 연산은 그림 2와 같이 외부 모드와 내부 모드로 구성된다. 내부 모드는 2단계로 이루어지고 외부 모드는 식 (2)와 같이 N 단계로 구성된다.

$$N = \log_2 |\text{Search Range}| + 1 \qquad (2)$$

본 논문에서는 탐색영역을 ±32로 가정한다. NTSS에서 내부 모드와 외부 모드의 동작은 많은 양의 메모리 접근을 필요로 하고 메모리 접근에 많은 사이클이 소요된다. 메모리 접근에 소요되는 시간을 줄이기 위해 탐색영역의 내부 메모리를 4개('SRAM1'~'SRAM4')로 분할하여 한 번에 여러 데이터를 동시에 읽을 수 있도록 하였다.

메모리에서 데이터를 읽는데 소요되는 사이클 수를 줄이기 위해 그림 6과 같이 처음 단계('Stage 1'~'Stage 3')에서는 최소값을 갖는 탐색점이 결정되기 전에 데이터를 메모리로부터 읽을 수 있게 하였다. 이 구간 동안 메모리에서 읽은 데이터는 SAD 계산과 병행하여 'Shared Buffer'에 저장된다. 'Shared Buffer'는 3개의 버퍼(NB1~NB3)로 구성된다. NTSS 내부 모드는 탐색영역이 ±1이므로 메모리 접근 시간을 줄이기 위해 내부 모드의 2개 단계의 데이터들을 버퍼 NB1에 담는다. 외부 모드의 'Stage 5'와 'Stage 6'은 ±1과 ±2의 탐색영역을 가지므로 이전과 현재 탐색 사이에 데이터의 중복이 발생한다. 따라서 버퍼 NB3에 데이터를 한꺼번에 저장하여 외부 메모리로부터 접근을 줄여 전체 연산 사이클 수를 줄였다. 표 3은 NTSS가 외부 모드로 동작할 때 필요한 단계의 수를 나타낸 것이다. 탐색영역이

표 3. 탐색영역에 따른 NTSS 단계 수
Table 3. Number of NTSS stages for various search ranges.

	탐색 영역			
	±64	±32	±16	±8
NTSS 단계 수	7	6	5	4

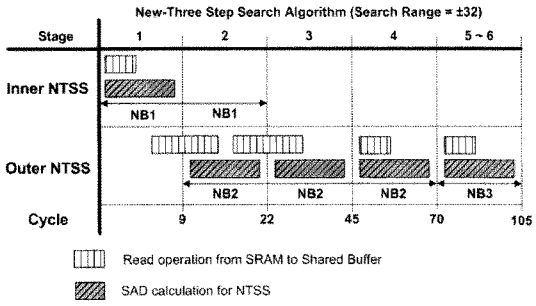

그림 6. NTSS 동작의 타이밍도
Fig. 6. Timing diagram of NTSS operations.

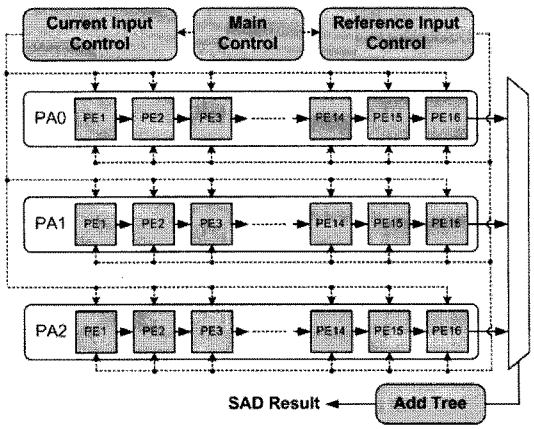

그림 7. 3단 파이프라인 움직임 추정 회로
Fig. 7. Three-stage pipeline motion estimation circuit.

커질수록 NTSS 연산에 소요되는 단계 수가 증가하게 되고 이에 따라 외부 메모리로부터 읽어 들이는 데이터의 양과 사이클 수가 증가하게 된다.

2. PE Array

4x4 블록 단위의 SAD를 구하기 위해 각 단계에서 9개의 탐색점 별로 16개씩, 총 144개의 PE(Processing Element)가 필요하다. 각 PE는 참조 데이터로부터 현재 데이터를 뺀 후 절대값을 구한다. 그림 7은 3단 파이프라인을 이용한 움직임 추정 회로의 블록도를 나타낸 것이다. 3단 파이프라인 스케줄링을 통해 PE를 16개씩 3개 그룹(PA0~PA2)으로 묶어서 48개로 줄였다. 9개 탐색점에 대한 스케줄링은 표 4에 나타내었다.

표 4. 파이프라인 스케줄링을 이용한 PE 연산
Table 4. PE operations using pipeline scheduling.

	탐색점			PE 수
	Cycle 1	Cycle 2	Cycle 3	
기존 방법	1,2,3,4,5,6,7,8,9	-	-	144
제안한 방법	1,2,3	4,5,6	7,8,9	48

3. SRAM 제어와 주소 생성

NTSS는 그림 4에서 보인 것 같이 4개의 SRAM을 이용한다. 'Stage 1'에서는 4개의 SRAM이 동시에 사용된다. 이후 단계들은 이전 단계에서 선택된 점에 따라 사용되는 SRAM이 달라진다. 표 5는 'Stage 2'와 'Stage 3'에 사용되는 SRAM 종류를 나타낸다. 'Stage 2'~'Stage 6'의 경우 SP(Search Point)가 2, 4, 6, 8인 경우를 제외하고 1개의 SRAM을 사용한다. 본 논문에서는 이전에 선택된 중심점의 위치를 3가지(center, vertical, horizontal)로 구분하여 효율적으로 제어할 수 있도록 설계하였다. NTSS 연산에 사용되는 SRAM 주소는 그림 8과 같은 방법에 따라 각 단계 별로 주소를 생성하여 다양한 탐색범위에 적용할 수 있도록 하였다.

4. 블록 모드 결정과 움직임 벡터 추정

앞 절에서 설명한 것과 같이 블록 모드 결정방법은 4x4 블록 단위로 NTSS를 수행한 후 생성된 16개 움직임 벡터를 이용하여 계산한다. 블록 모드를 결정하기 위해 모든 블록 크기(4x4, 8x4, 4x8, 8x8, 16x8, 8x16, 16x16)에 대하여 분산을 계산한다. 분산 연산에는 많은

표 5. SRAM의 사용 방법
Table 5. Usage of SRAM's.

SP	NTSS Stage		
	Stage 2 (*if psp is* 2)	Stage 3 (*if psp is* 2)	Stage 3 (*if psp is* 9)
1	SRAM1	SRAM1	SRAM1
2	SRAM1, SRAM2	SRAM1, SRAM2	SRAM2
3	SRAM2	SRAM2	SRAM2
4	SRAM1, SRAM3	SRAM1	SRAM2
5	Finish NTSS	Finish NTSS	Finish NTSS
6	SRAM2, SRAM4	SRAM2	SRAM2
7	SRAM3	SRAM1	SRAM2
8	SRAM3, SRAM4	SRAM1, SRAM2	SRAM2
9	SRAM4	SRAM2	SRAM2

psp: previous search point

SP	Address
1	Addr SP2 – (Search_range >> 2)
2	Start addr – (Search_range << 3)
3	Addr SP2 + (Search_range >> 2)
4	Start addr – (Search_range >>2)
5	Start addr
6	Start addr + (Search_range >>2)
7	Addr SP8 – (Search_range >> 2)
8	Start addr + (Search_range << 3)
9	Addr SP8 + (Search_range >> 2)

NTSS Stage	Search Range
1	32
2	16
3	8
4	4
5	2
6	1

Search Range = ± 32
SP = Search Point

그림 8. SRAM 주소의 생성
Fig. 8. Generation of SRAM addresses.

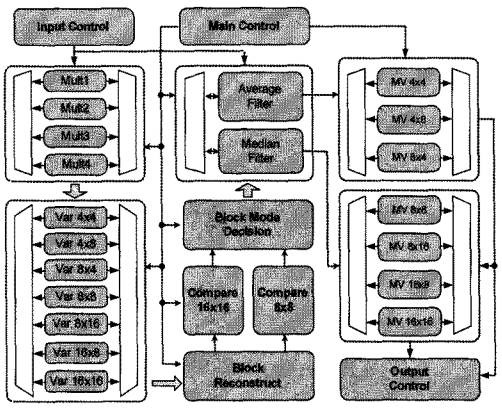

그림 9. 블록 모드 결정과 움직임 벡터 추정 회로의 구조
Fig. 9. Circuit architecture of block mode decision and motion vector estimation.

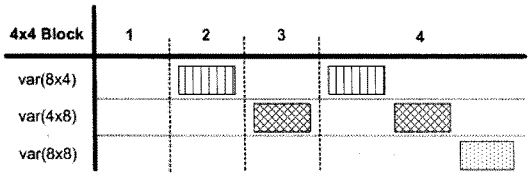

그림 10. 8x8 블록 모드 결정의 타이밍도
Fig. 10. Timing diagram of 8x8 block mode decision.

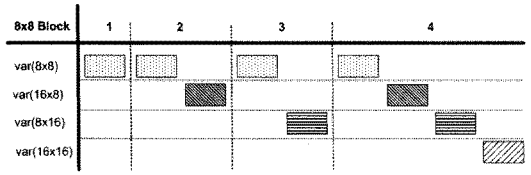

그림 11. 16x16 블록 모드 결정의 타이밍도
Fig. 11. Timing diagram of 16x16 block mode decision.

곱셈기가 사용되는데 적절한 스케줄링을 통해 4개의 곱셈기('Mult1'~'Mult4')만을 사용하였다. 그림 9는 블록 모드 결정과 움직임 벡터 추정 회로의 구조를 나타낸 것이다.

그림 10과 그림 11은 8x8 블록과 16x16 블록의 블록 모드를 결정하는 스케줄링 방법을 나타낸 타이밍도이다. 모든 블록 모드가 결정되면 각 블록 모드의 움직임 벡터 결정은 그림 9와 같이 3개 블록 모드(4x4, 4x8, 8x4)에 대해 'Mean Filter'를 이용하고 4개 블록 모드(8x8, 8x16, 16x8, 16x16)는 'Median Filter'를 이용한다.

5. 부화소 움직임 추정

H.264/AVC에서 부화소 보간은 1/2 화소와 1/4 화소에 대해 이루어진다. 1/2 화소는 정수 단위 움직임 추정에서 예측된 정수 화소로부터 보간하고, 1/4 화소는 1/2 화소와 정수 화소를 이용하여 보간한다. 1/2 화소를 보간 하는 경우 각 화소 당 9번의 보간 연산이 필요하고, 4x4 블록은 총 144(=16x9)번의 보간이 필요하다. 4x4 블록을 보간하는 경우[5] 그림 12와 같이 현재 블록과 다음 블록 간에 재사용이 가능한 화소들이 존재한다. 4x4 블록 단위로 보간하면 가로 방향 16번, 세로 방향 20, 대각선 방향 16번, 대각선 방향을 위한 추가연산 20번, 총 72번의 1/2 화소 보간이 필요하므로 정수 화소 단위로 1/2 화소를 보간 하는 경우보다 72번(50%)의 연산을 줄일 수 있다. 본 논문에서는 보간 회로의 성능을 높이기 위해 다음과 같은 5가지 방법을 통해 전체 연산 사이클 수를 줄였다.

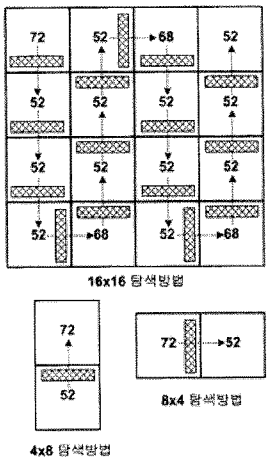

그림 12. 부화소 보간의 연산순서
Fig. 12. Calculation order for fractional interpolation.

1) 입력 화소의 재사용을 통한 메모리의 효율적사용
2) 덧셈기, 쉬프트 연산기로만 구현된 보간 필터
3) 4x4 블록 단위의 보간 연산을 통한 가변블록 처리
4) 스케줄링을 이용한 보간 필터 수의 제어
5) 2차원 탐색방법을 통한 부화소의 효율적인 재사용

그림 13에 나타낸 것과 같이 입력된 정수 화소는 'Input Register Bank'에 저장되고 다음 블록의 보간 연산에서 재사용하여 그림 12와 같이 외부 메모리 접근을 줄였다. 저장된 정수 화소는 그림 13의 오른쪽과 같이 덧셈기와 쉬프트 연산기로 구성된 휘도용 6-탭 FIR 필터와 색차용 필터에 전달되어 4x4 블록 단위로 1/2, 1/4, 1/8 보간 연산을 수행한다. 휘도 보간의 경우 1/2, 1/4 보간시 스케줄링을 통하여 3개의 6-탭 FIR 필터만을 사용하였고, 색차 보간의 경우 4개의 필터만을 사용하여 전체 회로 크기를 줄였다. 대각선 방향 1/2 보간 연산에 사용되는 재사용 가능한 1/2 화소들은 'Half-pel Register Bank'에 저장하여 다음 블록의 대각선 방향

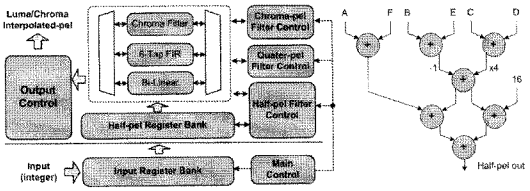

그림 13. 부화소 움직임 추정 회로와 필터 구조
Fig. 13. Fractional motion estimation circuit and filter structure.

보간 연산에서 재사용된다. 이러한 방법으로 얻어진 부화소들은 'Output Control' 회로를 거쳐 출력된다.

IV. 성능 분석 및 결과 비교

본 논문에서 제안된 회로는 Verilog HDL(Hardware Description Language)를 사용하여 RTL로 구현하였으며, Cadence사의 NC-Verilog를 이용하여 검증하였다. 회로의 동작은 시뮬레이션 결과와 레퍼런스 프로그램[6] 결과를 비교하여 RTL과 게이트 레벨에서 각각 검증하였다. Synopsys사의 Design Compiler를 통해 130nm 표준 셀 라이브러리를 이용하여 논리 수준 회로로 합성한 결과, 정수 화소 움직임 추정 회로는 표 6과 같이 77,600 게이트로 구성되었으며 최대 동작주파수는 161MHz이다. 합성된 회로는 D1(720x480) 칼라 영상을 초당 51장 처리할 수 있다. 기존에 연구된 방법들과 비교하면 표 7과 같이 48개의 PE를 이용하여 4x4 블록을 처리하는데 105 사이클이 소요되어 회로 크기나 처리

표 6. 정수화소 움직임 추정 회로의 합성 결과
Table 6. Synthesis results of integer-pixel motion estimation circuit.

회로 블럭	게이트 개수	최대지연시간 (ns)
Frame Selection	5,000	2.55
NTSS	14,800	5.65
Shared Buffer	27,000	5.69
Main Control	7,900	5.65
SRAM Control	7,800	2.69
Block Mode Decision	9,500	5.7
MV Estimation	5,600	5.7
회로 전체	77,600	5.7

표 7. 정수화소 움직임 추정 회로의 성능 비교
Table 7. Performance comparison of integer-pixel motion estimation circuits.

	[7]	[8]	[9]	[10]	제안한 방법
PE 수	9	16	16	18	48
게이트 수	36,600	30,000	22,300	-	14,800
탐색 영역	±7	±7	±16	±7	±32
사이클 수	794	324	683	256 ~512	105

표 8. 부화소 움직임 추정 회로의 성능 비교
Table 8. Performance comparison of fractional motion estimation circuits.

	[12]	[13]	제안한 방법
공정(nm)	180	180	130
최대 동작주파수 (MHz)	143.5	148.5	200
게이트 수	17,168	21,569	22,478
영상 해상도	1,280x720	1,920x1,088	1,920x1,088
처리속도(fps)	30	60	69
처리시간 (cycles/MB)	492	600	492
지원 블록크기	4x4	All	All
지원 휘도/색차	휘도	휘도/색차	휘도/색차

속도 면에서 우수한 성능을 갖는 것을 확인할 수 있다. 18개의 PE를 이용한 병렬 처리를 통해 원래 NTSS 보다 회로의 탐색 속도를 1/2 가량 줄이는 논문 [11]과 비교하는 경우에도 처리 속도 면에서 우수한 성능을 갖는다. 부화소 움직임 추정 회로는 표 8과 같이 22,478 게이트로 구성되었으며 최대 동작주파수는 200MHz이다. 제안된 방법을 이용하여 구현한 회로는 1080HD (1,920x1,088)급 칼라 영상을 초당 69장 까지 처리할 수 있다.

V. 결 론

H.264/AVC의 움직임 추정 과정에서 발생하는 복잡한 연산과 압축에 소요되는 시간을 줄이기 위해 빠른 다중 참조 프레임 선택, 분산을 이용한 가변 블록 모드 결정, 움직임 벡터 예측을 고속으로 처리하는 방법을 사용하였다. 이 방법들을 기반으로 효율적인 메모리 사용 및 파이프라인 구조, 스케줄링을 통한 효율적인 NTSS 적용을 통하여 고성능 정수 화소 움직임 추정 회로를 구현하였다. 또한 움직임 추정에서 많은 연산량을 차지하는 부화소 보간을 입력 데이터의 재사용을 통한 메모리의 효율적 사용, 4x4 블록 단위의 부화소 보간 연산, 계산된 부화소의 재사용, 스케줄링을 통한 보간 필터 사용, 덧셈기와 쉬프트 연산기만을 이용한 6-탭 FIR 필터 구현 등을 통해 H.264/AVC용 고성능 부화소 움직임 추정 회로를 구현하였다. 설계한 회로는 130nm 표준 셀

공정을 사용하는 경우, 정수 화소 단위 움직임 추정 회로는 D1급 칼라 영상을 1초당 51장 까지 처리할 수 있고, 부화소 단위 움직임 추정 회로는 1080HD급 칼라 영상을 1초당 69장 까지 처리할 수 있다.

참 고 문 헌

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

[2] R. Li, B. Zeng, and M.L. Liou, "A new three-step search algorithm for block motion estimation," IEEE Trans. on Circuits and Systems for Video Technology, vol. 4, issue 4, pp. 438-442, Aug. 1994.

[3] C. Ting and L. Po, "Center-biased frame selection algorithms for fast multi-frame motion estimation in H.264," Proceedings of the 2003 International Conference on Neural Networks and Signal Processing, vol. 2, pp. 1258-1261, Dec. 2003.

[4] P. Yin, H.-Y.C. Tourapis, A.M. Tourapis, and J. Boyce, "Fast mode decision and motion estimation for JVT/H.264," International Conference on Image Processing, vol. 3, pp. 853-856, Sep. 2003.

[5] C. Yang, S. Goto, and T. Ikenaga, "High performance VLSI architecture of fractional motion estimation in H.264 for HDTV," IEEE International Symposium on Circuits and Systems, pp. 2605-2608, May 2006.

[6] JVT H.264 Reference Software Version JM11.

[7] H-M. Jong, L-G. Chen, and T-D. Chiueh, "Parallel architectures for 3-step hierarchical search block-matching algorithm," IEEE Trans. on Circuits and System for Video Technology, vol. 4, no. 4, pp. 407-416 Aug. 1994.

[8] Y.S. Jehng, L.G. Chen, and T.D. Chiueh, "A motion estimator for low bit-rate video CODEC," IEEE Trans. on Consumer Electronics, vol. 38, issue 2, May 1992.

[9] P.M. Kuhn, Algorithms, complexity analysis and VLSI architectures for MPEG-4 motion estimation, Kluwer Academic Publishers, 1999.

[10] K. Seth, P. Rangarajan, S. Srinivasan, V. Kamakoti, and V. Bala Kuteshwar, "A parallel architectural implementation of the New Three-Step Search algorithm for block motion estimation," International Conference on VLSI Design, pp. 1071-1076, Jan. 2004.

[11] M. Ho, J. Huang, S. Chin, and C. Hsu, "High efficient NTSS-based parallel architecture for motion estimation in H.264," International Conference on Communications, Circuits and Systems, pp. 679-683, May 2008.

[12] M. Li, W. Ronggang, and W. Wu, "The high throughput and low memory access design of sub-pixel interpolation for H.264/AVC HDTV decoder," IEEE Workshop on Signal Processing Systems Design and Implementation, pp. 296-301, Nov. 2005.

[13] J. Zheng, W. Gao, D. Wu, and D. Xie, "A novel VLSI architecture of motion compensation for multiple standards," IEEE Trans. on Consumer Electronics, vol. 54, issue 2, pp. 687-694, May 2008.

저 자 소 개

이 선 영(학생회원)
대한전자공학회 논문지
제45권 SD편 제9호 참조

조 경 순(평생회원)
대한전자공학회 논문지
제45권 SD편 제9호 참조