반화소 단위의 움직임 정보 추정을 위한 VLSI 구조에 관한 연구

황두영*, 이승원**, 이동호*

**동양컴퓨터 기술개발(주), *한양대학교 제어계측공학과

A VLSI Architecture for Motion Estimation with Half-pel Resolution

Doo-Young Hwang, Sung-Won Lee*, Dong-Ho Lee**

*Oriented Computer Technology Development Co., Ltd.

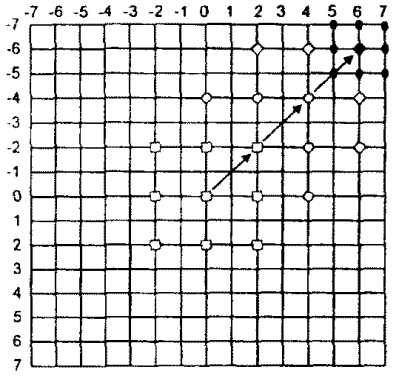
**Dept. of Control & Instru. Engr., Hanyang University

Ⅰ. 서론

최근의 동영상 압축기술은 멀티미디어 통신, 비디오 폰, 원거리 화상회의, HDTV, CD-ROM 등 그 응용 분야가 점차 확대되고 있다. 영상 압축 기술의 핵심은 연속되는 비디오 신호에 생길 수 있는 시간적, 공간적 정보에 대한 중복성 이용에 있다. 여러가지 압축 기법 중에서 동영상 움직임 보상 부호화 기법이 많이 이용되고 있다. 움직임 보상 부호화에는 움직임 추정에 의해 움직임을 보상하는 부분과 예측 오차를 부호화 하는 부분으로 이루어져 있다. 움직임 벡터 추정은 움직임 정보의 유사성을 고려하여 블럭 정합 알고리즘(BMA: block matching algorithm)이 가장 많이 이용되고 있다.

블럭 정합 알고리즘에는 FS(Full Search Algorithm)[1], TSS(Three Step Search Algorithm)[2], HS(Hierarical Search Algorithm)[3], 4SS(Four Step Search Algorithm)[4]등 많은 알고리즘 등이 논문으로 발표되고 일부 알고리즘은 실제로 구현되어 쓰이고 있다. 이중 FS 는 가장 성능은 좋지만 많은 연산량이 필요로 하므로 실시간 구현에 어려움이 있고 TSS 는 통계적인 움직임 벡터를 고려하지 않고 모든 움직임 벡터의 가능성을 같게 보기 때문에 실제 영상에서는 비효율적인 면을 나타낸다. 그 반면 4SS 는 FS 만큼의 많은 양의 연산수를 요구하지 않으면서 TSS 보다는 높은 효율성을 가지는 것으로 알려져 있다[4].

따라서 본 논문에서는 실시간 움직임 벡터 추정기에 4SS 알고리즘을 이용하고 선형 근사 방식 반화소 추정법을 사용한 반화소 단위의 움직임 벡터 추정기의 VLSI ARCHITECTURE 를 제안하고자 한다.



<그림 1. 4SS search path >

Ⅱ. 4SS(Four Step Search Algorithm)소개

일반적으로 거의 대부분의 영상은 연속해서 이어지며 천천히 움직인다. TSS 에서는 첫번째 단계에 search window 를 균일하게 checking point 들을 잡기 때문에 움직임이 적은 일반 영상에서는 비효율적인 면을 보인다. 이에 4SS 에서는 통계적으로 일반적인 움직임의 양이 중심에서 -2 ~ 2 화소 정도의 움직임을 보임을 바탕으로 하여 처음 중심에서 가로 세로 각각 -2, 2 떨어진 9 점에서 searching 을 시작한다. 때문에 4SS 에서는 움직임이 비교적 적은 영상에도 잘 적응한다. 4SS 처리 순서는 다음과 같다.

STEP 1. 15 * 15 searching area 의 중심에서 5*5 window 에 9 개의 checking point 에대해 최소 SAD 값을 갖는 점을 찾는다.

STEP 2. STEP1 에서 구해진 점을 중심으로 5*5 window 에 9 개의 checking point 에 대해 최소 SAD 값을 갖는 점을 찾는다.

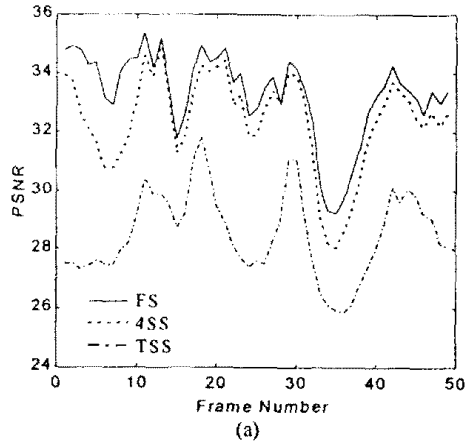
STEP3. STEP2 에서 구해진 점을 중심으로 2 와 같은 과정을 거친다.

STEP4. 3 에서 구해진 점을 중심으로 3*3 window 에 9 개의 checking point 에 대해 최소 SAD 값을 갖는 점을 찾는다. 여기서 찾아진 점이 움직임 벡터의 방향을 결정하게 된다.

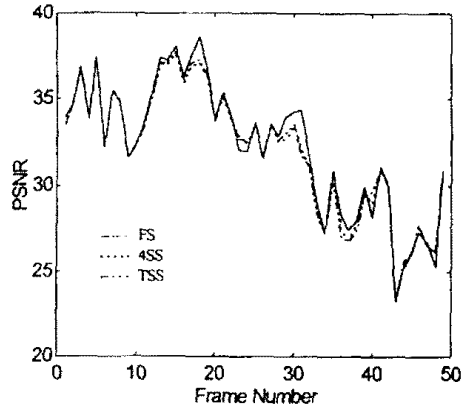
그림 2. (a), (b)는 FS, TSS, 4SS 를 table tennis 영상과 susi 영상에서 비교한 것이다. 704*480 해상도의 table tennis 영상은 한 사람이 탁구를 치고 있는 모습인데 사람과 공만 움직이고 배경은 고정된 상태이다. 그림 2(a)에서 보는 바와 같이 이 영상에서는 TSS 의 PSNR 이 FS 나 4SS 에 비하여 많이 떨어지고 4SS 의 PSNR 은 FS 에 거의 근사한 모습을 볼 수 있다. 그것은 이 영상에서는 움직임이 없는 부분이 화면의 상당부분을 이루고 있어서 TSS 에 비하여 4SS 가 움직임이 다소 적은 화면에서 더 잘 적응하기 때문이다.
352 * 240 해상도를 갖는 susi 영상은 한 여자가 수화기를 귀에 대고 있다가 머리를 뒤로 넘기는 모습이다. 이 영상은 화면의 중앙에 얼굴이 크게 나와 화면내 움직임이 -2 - 8 사이에 골고루 분포하기 때문에 그림 2(b)와 같이 FS, TSS, 4SS 가 거의 근사한 PSNR 곡선을 그린다. 또 그림 2. (a), (b)에서 PSNR 곡선이 밑으로 많

이 내려온 부분은 회전을 하거나 모양체가 변하기 때문에 BMA(Block Matching Algorithm)를 사용했기 때문에 생기는 현상이다.

이 시뮬레이션 결과에서는 4SS 는 TSS 보다는 성능이 월등하고 FS에 거의 접근한다는 것을 알 수 있다.



(a)

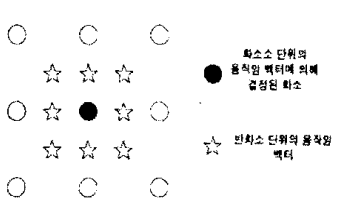


(b)

< 그림 2. (a) "table tennis" sequence, (b) "susi" sequence 에 대한 FS, TSS, 4SS 의 PSNR 비교>

Ⅲ. 반화소 단위의 움직임 추정

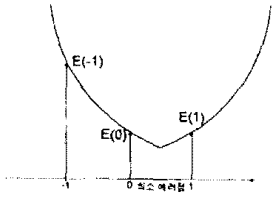
3.1. 보간법에 의한 전 영역 탐색 반화소 움직임 벡터 결정법.



<그림 3. 보간법에 의한 반화소 >

각 반화소의 움직임 벡터에서의 움직임 보상 에러치를 보간법에 의해 구한 다음 그 값들을 비교하여 가장 작은 경우로 반화소 단위의 움직임 벡터를 구한다. 이 방법에서는 움직임 벡터에 의해 정해진 화소 근처에 8 점의 반화소에 대해 모두 비교하기 때문에 성능은 좋지만 연산수가 많아진다는 단점이 있다.

3.2. 포물선 근사화 방식



< 그림 4 포물선 근사화 방식>

$E = ax^2 + bx + c$ (a>0)
$E(-1) = a - b + c$
$E(0) = c$
$E(-1) = a + b + c$

위의 그림에서 error 곡선은 양쪽이 대칭인 포물선이라고 가정했으므로 최소 에러점이 -0.75~-0.25 사이에 있는 경우에는

$3\{(E(-1) - E(0)\} < (E(1) - E(0)$ ----- (1)

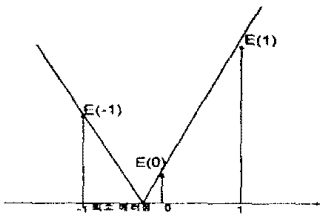
이고 최소에러점이 0.25~0.75 사이에 있는 경우에는

$(E(-1) - E(0) > 3\{(E(1) - E(0)\}$ ----- (2)

이다.

(1) 식의 경우에는 mv-0.5, (2)식의 경우에는 mv+0.5 를 하여 반화소 단위의 움직임 정보를 얻게 된다.

3.3. 선형 근사화 방식



<그림 5. 선형 근사화 방식>

$E = a|x - b| + c$ a: 기울기, b:최소점
$a > 0, |b| < 1$

$E(-1) = a(1 + b)$
$E(0) = a|b|$
$E(1) = a(1 - b)$

위의 그림에서 error 곡선은 양쪽이 대칭인 직선이라고 가정했으므로 최소 에러점이 -0.57~-0.25 사이에 있는 경우에는

$2\{(E(-1) - E(0)\} < (E(1) - E(0)$ ----- (3)

이고

최소에러점이 0.75~0.25 사이에 있는 경우에는

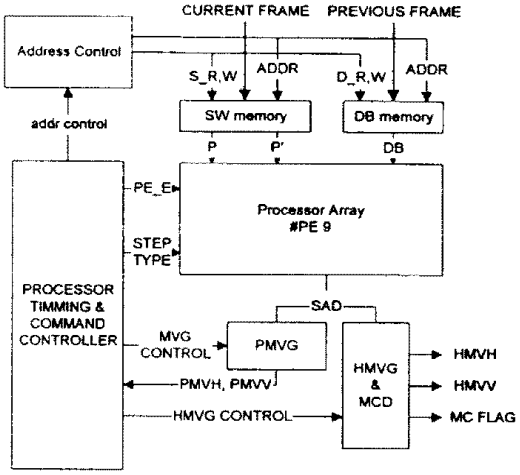
$$(E(-1) - E(0) > 2\{(E(1) - E(0)\} \quad \text{----- (4)}$$

이다.

(3) 식의 경우에는 mv-0.5, (4)식의 경우에는 mv+0.5 를 하여 반화소 단위의 움직임 정보를 얻게 된다.

IV. VLSI ARCHITECTURE

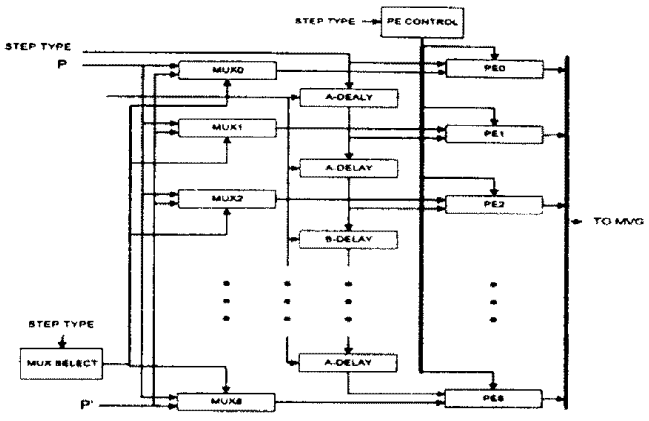
본 논문에서 제안하는 VLSI 구조는 기존에 제안된 4SS 알고리즘을 적용하고 반화소 단위의 움직임을 추정하기 위해 STEP 수를 1 개 첨가하여 5 번째 STEP 에서 반화소 단위의 움직임 추정을 하고 MC, no MC 를 결정할 수 있다. 반화소 단위의 움직임 추정에는 성능과 연산 면에서 적절한 선형 근사화 방식을 사용하였다..



<그림 6. 반화소 움직임 추정기의 전체 블락도>

그림 6 은 반화소 단위 움직임 추정기의 전체 블럭도이다. 전체 구조는 Search Window 와 기준 블럭의 메모리를 제어하고 어드레스 생성하는 Address Control 부분과 SAD(Sum of Absolute Difference) 계산을 위한 9 개의 Processor element 로 구성된 Processr Array, 움직임 벡터를 계산하고 반화소 단위의 움직임 추정을 하고 움직임 보상 flag 를 발생시키는 PMVG(Preliminary Motion Vector Generator)와 HMVG(Half pel Motion Vector Generator) & MCD(MC Decision), 전체 블락의 동작 timming 을 맞추고 step type 을 결정하는 processor controller 부분으로 구성되어 있다. PMVG 에서는 9 개의 PE(Processor Element)에서 출력된 SAD 값을 비교하여 그중 최소 에러값을 갖는 부분을 다음 STEP 에서 수행될 9 점에 대한 INDEX 로 넘겨주는 역할을 한다.

그림 7 은 움직임 추정기 내에 있는 프로세서 어레이의 내부 구조이다. SAD 를 계산하기 위한 9 개의 PE 가 있고 Search Window 의 상위 데이타 P, 하위 데이타 P'를 분배해주는 mux 가 9 개 있고 각 mux 에 대한 select 신호를 만들어 주는 mux select generator 가 있고, delay 수가 다른 두가지 A-Delay, B-Delay 가 있고 각 Delay 소자는 step 의 형태에 따라서 두 가지 delay 형태를 갖는다



< 그림 7. 프래세서 어레이의 구조>

Step type 은 4SS 에서 1,2,3 step 과 4, 반화소 단위의 움직임 계산을 위한 5step 두가지 형태를 분리하는데 1,2,3 step 에서는 A-Delay, B-Delay 가 각각 2, 24cycle delay 되도록 선택하고 4step 과 5step 에서는 1,12 cycle delay 가 되도록 선택해 준다. 또 이 step type 에 따라 mux select generator 의 출력도 달라지게 된다.

< 표 1. step 1,2,3 일때 PE enable signal timming >

CLK	PE 0	PE 1	PE 2	PE 3	PE 4	PE 5	PE 6	PE 7	PE 8
0-1	1	0	0	0	0	0	0	0	0
2-3	1	1	0	0	0	0	0	0	0
4-31	1	1	1	0	0	0	0	0	0
32-33	1	1	1	1	0	0	0	0	0
34-35	1	1	1	1	1	0	0	0	0
36-63	1	1	1	1	1	1	0	0	0
64-65	1	1	1	1	1	1	1	0	0
66-67	1	1	1	1	1	1	1	1	0
68-323	1	1	1	1	1	1	1	1	1

< 표 2 step 4,5 일때 PE enable signal timming >

CLK	PE 0	PE 1	PE 2	PE 3	PE 4	PE 5	PE 6	PE 7	PE 8
0	1	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0
2-15	1	1	1	0	0	0	0	0	0
16	1	1	1	1	0	0	0	0	0
17	1	1	1	1	1	0	0	0	0
18-31	1	1	1	1	1	1	0	0	0
32	1	1	1	1	1	1	1	0	0
33	1	1	1	1	1	1	1	1	0
34-289	1	1	1	1	1	1	1	1	1

표 1 과 표 2 는 각 step 에 대한 9 개의 PE 의 동작 enable 신호이다. 각 PE 에 대해 enable 신호가 1 이 되면 SAD 계산을 하게 되고 0 이 되면 동작을 하지 않는다. Step 1,2,3 에서는 PE0,PE1,PE2 사이, PE3,PE4,PE5 사이, PE6,PE7,PE8 사이에서는 1 cycle 씩 delay 된 다음 동작하고 PE2, PE3 사이, PE5,PE6 사이에서는 14 cycle 씩 delay 된 다음 동작을 하게 된다.

또 Step 4,5 에서는 PE0,PE1,PE2 사이, PE3,PE4,PE5 사이, PE6,PE7,PE8 사이에서는 2 cycle 씩 delay 된 다음 동작하고 PE2, PE3 사이, PE5,PE6 사이에서는 28 cycle 씩

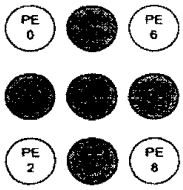
delay 된 다음 동작을 하게 된다.

각 PE에 입력되는 데이타 값들은 A-Delay, B-Delay 에 의해 각각 step 1,2,3 에서는 1 cycle, 2 cycle, step 4,5 에서는 14cycle, 28cycle 씩 delay 된 값들이 입력되게 된다.

<표 3. Half pel estimation timming>

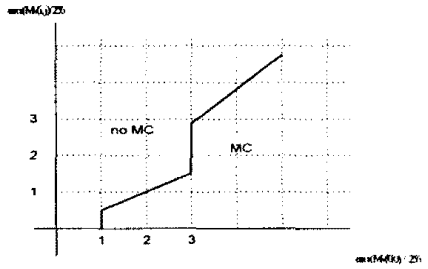
CLK	PE 0	PE 1	PE 2	PE 3	PE 4	PE 5	PE 6	PE 7	PE 8
259	x	O	x	x	x	x	x	x	x
274	x	x	x	O	x	x	x	x	x
275	x	x	x	x	O	x	x	x	x
276	x	x	x	x	x	O	x	x	x
291	x	x	x	x	x	x	x	O	x

표 3 은 반화소 단위 움직임 벡터 계산을 위해 step 5 에서 각 PE 로부터 SAD 출력이 나오는 timming 이다.



<그림 8. Half pel 계산을 위한 PE 들>

그림 8 과 같이 half pel estimation 을 하기 위해서는 PE1, PE3, PE4, PE5, PE7 이상 5 개의 SAD 값이 필요하다. 표 3 과 같은 timming 대로 각 PE 의 출력이 나오면 출력된 SAD 값이 나오는 시각까지 데이타를 저장하고 있다가 비교할 PE 의 출력이 나올때 식 (3)과 (4)와 같이 반화소 움직임 계산식에 의해 반화소 움직임 벡터를 계산을 하게 된다.



<그림 9. 움직임 보상 결정 곡선 >

이렇게 반화소 단위 움직임 추정까지 끝나면 단화소 단위 움직임 벡터와 함께 그림 9 의 움직임 보상 결정 곡선에 의해 MC 인지 no MC 인지 결정을 하여 MC flag 신호로 내보내게 된다.

본 논문에서 제안하는 구조에서는 동작에 소요되는 총 clk 수는 1565 cycle 로 그 계산과정은 다음과 같다.

STEP 1,2,3 에서 (256+68+2)*3 = 978
STEP 4 에서 256+34+2 = 292.
STEP 5 에서 256+33+2 = 291
HMV 계산 2
MC 결정 2

V. 결론

본 논문에서는 FSS(Four Step Search Algorithm)을 이용하여 실시간 시스템으로 구현될 수 있도록 동작에 소용되는 클럭수를 줄였으며 1 step 을 추가하여 반화소 단위의 움직임 추정과 움직임 보상결정까지 할 수 있는 VLSI 구조를 제안하였다.

FSS, TSS, 4SS 의 시뮬레이션 결과에 의해 TSS 보다 성능이 좋고 FSS 보다 적은 수의 클럭수를 요구하는 4SS 를 이용하였으므로 실시간 시스템에서 적용하기에 적합한 구조이다.

본 논문에서 제안하는 구조로 움직임 추정기를 사용하였을 경우, 6.5MHz 의 속도로 영상 데이타가 입력될 경우 약 26MHz 로 시스템을 동작시킬 수 있다.

또 본 논문에서 제안하는 구조는 Processor Array 를 2 개를 사용하고 2 배의 입력 메모리를 사용하면 search window 를 확장시켜 MPEG2, HDTV 에서도 사용할 수 구조로 확장시킬 수 있다.

< 참고 논문 >

[1]T.Komarek and P. Pirsch, "Array architectures for block-matching algorithms," IEEE Trans. Circuits Syst., vol. 36, Oct. 1989..

[2].Kun -Win Yang, Ming-Ting Sun, and Lamcelpt WU, " A Family of VLSI Designs for the Motion Compensation Block-Matching Alforithm". IEEE Trans. Circits Syst. Vol36. OCT. 1989.

[3]. L. De. Vos, "VLSI architectures fo th hierarchical block-matching algorithm," 1994 IEEE Int. Symp. Circuits Syst., vol. 4, pp 215-218.

[4] Lai-Man Po and Wing-Chung Ma. "A Novel Four-Step Search Algorithm for Fast Block Motion Estimation"., IEEE Trans. Circuits and Systems for Video Tech. Vol6. June 1996.

[5]. Santanu Dutta, Wayen Wolf, "A Flexible Parallel Architecture Adapted to Block-Matching Motion-Estimation Algorithms", IEEE Trans, Circuits Syst. Vol6. FEBRUARY 1996.