효율적인 SAD 연산을 위한 하드웨어 구현에 대한 연구

*박장호 **최현준 ***박성호 ****서영호 *****김동욱

광운대학교

*jhchip@kw.ac.kr

A study on Hardware Implementation for efficient SAD calculation

*Park, Jang-Ho **Choi, Hyun-Jun ***Park, Sung-Ho

****Seo, Young-Ho *****Kim, Dong-Wook

Kwang Woon University

요약

본 논문은 움직임 추정(motion estimation)과정에서 SAD(Sum of Absolute Difference)값을 추출하기 위해 사용되는 SAD 연산기의 게이트 수를 줄이는데 초점을 두고 하드웨어를 구현하였다. 게이트 수(gate count)를 줄이기 위한 방법으로 1의 보수(one's complement)의 수 체계를 이용하였다. 하드웨어 구현 결과, 게이트 수를 약 12%~25% 줄일 수 있었다.

1. 서론

최근 정보화 시대로의 도약과 함께 정보화 기술의 급속한 발달이 계속되고 있다. 특히 멀티미디어 서비스에 대한 수요와 공급은 기하급수적으로 증가하고 있으며, 다양한 네트워크 환경에서의 효율적인 멀티미디어 서비스를 위하여 우수한 비디오 압축 기술에 대한 요구가 어느 때보다 커지게 되었다[1]. JVT(Joint Video Team)라 불리는 비디오 압축 전문가 위원회에서는 H.264/AVC(Advanced Video Coding)로 알려진 표준화를 진행해왔다[2].

H.264/AVC는 기존의 MPEG-2/3/4의 압축 기술에 비해 압축 효율이 증가한 기술들을 선보이고 있다. 동영상 압축 코덱에서의 가장 중요한 핵심기술이면서, 가장 많은 부분을 차지하고 있는 부분은 움직임 추정이라고 할 수 있다. H.264/AVC는 움직임 추정 과정에서 16×16부터 4×4까지의 총 7가지의 다양한 가변 블록을 사용 한다. 또한, 움직임 추정을 수행할 때 정수 단위 화소 뿐만 아니라, 1/4 화소 단위까지 탐색을 하며 예측에 사용되는 참조 프레임도 한 장의 프레임에서 다중 프레임까지 사용함으로써, 보다 정확한 움직임 추정을 수행한다. 이러한 기존의 기술들보다 우수한 성능을 보여주고 있지만 높은 압축률과 고화질을 유지하기 위해서는 반복 연산을 해야 하므로 계산량이 많고 복잡하여 하드웨어 구현이 어렵다. 이러한 점들을 해결해보고자 많은 사람들이 탐색 지점(search point)을 줄이기 위해 TSS(Three-Step Search), NTSS(New Three-Step Search), DS(Diamond Search)등과 같은 고속 움직임 추정 방법(fast search algorithm)들의 대한 연구를 계속 진행해 왔다. 또한, 주변 블록의 움직임 벡터(motion vector)의 상관성(correlation)을 이용하여 탐색 시간을 크게 줄인 알고리듬들도 연구되었다[3][4][5].

움직임 추정 기법은 BMA(Block Matching Algorithm)과정을 통해 SAD 연산기를 이용하여, 최적의 예측 블록을 찾아낸다. SAD 연산기는 움직임 추정 기법이 끝날 때까지 반복적으로 사용되는 것으로 하드웨어 구현 시, 게이트 수를 증가시키는 원인이 된다. 게이트 수가 증가하면 칩의 크기(chip size)도 커지고 많은 전력 소모가 발생 한다.

본 논문에서는 게이트 수를 줄이기 위하여 1의 보수의 수 체계를 이용하는 방법을 제안하였다. 본 논문은 Ⅱ장에서는 BMA 과정에서 SAD 연산기를 이용하여 SAD값을 추출하는 방법을 소개하며, Ⅲ장에서 SAD 연산기를 하드웨어로 구현 시, 게이트 수를 줄일 수 있는 방법을 제안한다. Ⅳ장에서 제안된 방법을 적용하여 여러 가지 조건에 따라 실험을 해본 결과를 설명하며 Ⅴ장에서는 결론을 내리면서 끝맺음을 한다.

2. BMA(Block Matching Algorithm) 연산 과정

H.264/AVC의 예측 방법은 인트라 예측(intra prediction)과 인터 예측(inter prediction)으로 나눌 수 있다. 전체 부호화를 나타낸 그림 1에서 인터 예측은 움직임 추정(ME, motion estimation)과 움직임 보상(MC, motion compensation)부분이다. 인터 예측은 전체 부호화 과정에서 가장 중요한 부분이며 많은 관심을 받는 부분이기도 하다. 인터 예측을 어떻게 하느냐에 따라 영상의 화질 차이와 압축률에 영향을 끼치기 때문이다.

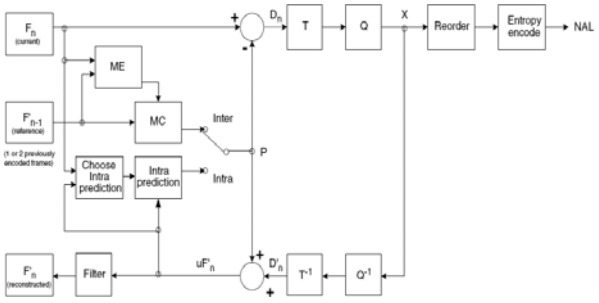

그림 1. H.264/AVC 부호화기의 구조

움직임 추정 기법에서는 그림 2에 나타낸 것처럼 블록 정합 알고리듬인 BMA방법을 사용한다. BMA는 이전 프레임으로부터 블록 단위로 움직임 추정을 하는 기법으로, 현재 프레임을 7가지의 다양한 크기의 블록으로 나누어 현재 프레임의 각각의 블록을 그에 해당하는 이전 프레임의 탐색 영역(search window)내의 블록들과 비교하여 가장 유사한 블록을 찾아내는 것이다.

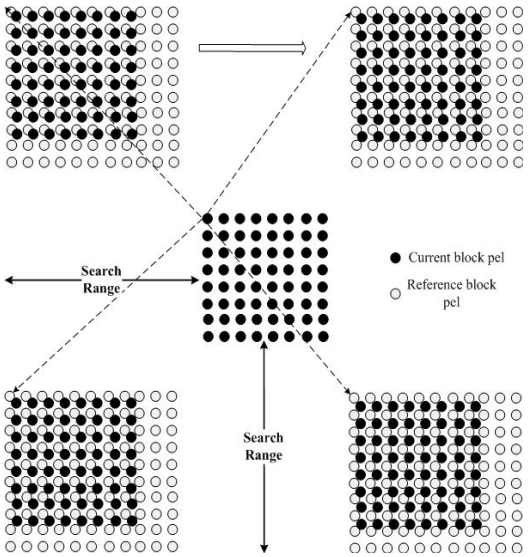

그림 2. 블록 정합 알고리듬 (Block Matching Algorithm)

BMA에서 유사한 블록을 찾기 위해 사용되는 평가 함수에는 MSE(Mean Square Error), MAD(Mean Absolute Difference), SAD(Sun Absolute Difference)등의 함수가 있다. 영상의 크기에 따라 정확성의 차이가 있을 수 있지만, 연산량의 효율성은 SAD가 가장 우수하다.

H.264/AVC에서 움직임 추정 부분이 전체 부호화 과정에서 가장 많은 부분을 차지하기 때문에 연산량이 가장 효율적인 SAD를 사용하며 식 (1)과 같이 나타낸다.

$$SAD = \sum_{i=0}^{M-1}\sum_{j=0}^{N-1} |A(a_{i,j}) - B(b_{i,j})| \quad (1)$$

A와 B는 프레임의 블록들을 나타내며 i, j는 화소를 나타낸다. MxN 블록 사이즈로 예측한다면, 그 현재 블록의 (0 ,0)의 위치와 참조 블록의 (0 ,0), (0, 1), (0, 2), (0 ,3)등의 위치와 매칭(matching)을 하면서 SAD 연산기를 이용하여 화소들의 차이 값(difference)을 합산해서 현재 블록과 참조 블록간의 가장 작은 SAD값을 구한다. 가장 작은 SAD값이 가지는 블록이 가장 유사한 블록으로 선택 된다. M×N 블록에서 SAD값을 추출하는 과정을 그림 2와같이 나타내었다.

MPEG-2와 MPEG-4에서는 움직임 예측시 실제 화소와 가장 근사한 화소를 찾고 이를 보간 한다, 그러면 가상의 화소가 만들어 지는데 이 화소들에 대해서 한번 더 움직임 예측을 수행한다. 이리하여 1/2 화소 단위까지의 움직임 추정을 사용한다. 하지만, H.264/AVC에서는 1/4 화소 정밀도까지 사용 한다. 또한 필요에 딸 1/8 화소 정밀도 까지 사용하기도 한다. 정수화소 이하의 예측신호를 생성하기 위해 6탭 FIR 필터를 사용한다.

이렇게 추가된 기능들 때문에 H.264/AVC는 기존의 압축 방식보다 우수한 기능을 보이는 것이다. 그러나 그만큼 복잡도가 더해져서, 비용과 하드웨어 구현시, 여러 가지 어려움이 생기고, 많은 하드웨어 자원소모량도 증가하였다.

3. 제안한 SAD 연산 구현 방법

H.264/AVC는 전 영역 움직임 추정 기법(full search)과 고속 움직임 추정 기법(fast search)을 사용한다. 어떤 움직임 추정 기법에서라도 SAD 연산기는 빼놓을 수 없는 중요한 역할을 한다. SAD값에 의해 가장 유사한 블록을 선택하기 때문이다. 그러나 SAD값만을 100% 신뢰 할 수 없다. 특히 고속 움직임 추정 기법에서는 가장 유사한 포인트의 선정이 잘못 선택 되어진다면, 움직임 추정은 local minimum 이라는 오류에 빠지게 된다. 또한, 유사한 블록을 찾아서 현재 프레임 예측에 사용했을 경우, 영상 화질이 현저하게 떨어질 경우에는 I_frame을 예측에 사용함으로써 움직임 추정에 정확성을 향상시킨다. 움직임 추정에서 SAD값을 찾을 때, 사용되는 SAD 연산기의 구조를 그림 4와 같이 나타낼 수 있다.

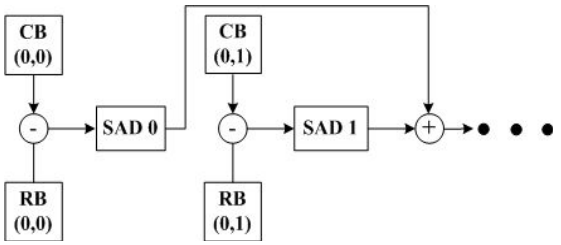

그림 4. SAD 연산기 구조

CB는 현재 블록(current block)을 의미 하고, RB는 참조 블록(reference block)을 의미 한다. 위의 그림에서 설명한 것과 같이 SAD 연산기에는 감산 연산이 들어가게 된다. 4x4 블록에서 최소 16개의 연산기부터 16x16 블록에서의 최대 256개의 연산기가 사용 된다 . 감산 연산은 하드웨어 여건상 부득이하게 보수(complement)를 사용할 수 밖에 없다. 음수를 표현할 때 1의 보수(one's complement)의 수 체계를 사용하면 음수 표현이 간단하지만 오버플로 처리로 연산과정이 복잡해질 수 있다. 반면, 2의 보수(two's complement)를 사용한다면 연산에서 오버플로를 무시하므로 연산이 간단해지는 장점이 있다.

그러나 이 방법은 음수 표현시 1의 보수를 취한 후, "+1"의 연산이 필요하다. SAD 연산은 음수 표현이 필요하므로, 1의 보수의 수 체계를 사용 하면 게이트 수가 줄 수도 있다는 가설을 세워 하드웨어 구현을 하고, 비교해 보았다. 표 1은 블록 사이즈를 16×16으로 가정 했을 시, 한 개의 SAD 연산기를 2의 보수(two's complement)와 1의 보수(one's complement)의 수 체계로 하드웨어로 구현을 하였고, 0.25μm 공정으로 합성을 하고 결과를 살펴보았다. 표 1은 두 가지 방법의 게이트 수를 보여 주고 있다.

표 1. Gate count

	1's complement	2's complement
Gate count	116	148

위의 표를 보면 알 수 있듯이, 1의 보수의 수 체계를 사용한 하드

웨어 구현이 2의 보수의 수 체계를 사용한 방법보다 많은 수의 게이트 수를 줄였다. 블록 사이즈마다 연산기의 수가 다를 것이다. 그림 5와 같이 16x16 블록사이즈로 탐색을 한다면 256개의 SAD 연산기가 필요하다. 탐색 영역이 넓어지거나 블록 사이즈가 커진다면 게이트 수는 더 많이 늘어날 것이다. Ⅳ장에서 본 논문에서 제안한 1의 보수의 수 체계 방식으로 하드웨어 구현을 하여, 여러 상황을 가정하여 실험을 해보았다.

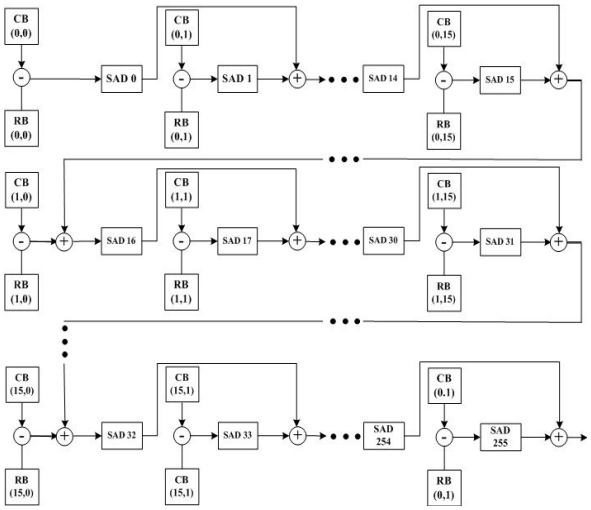

그림 5. Macro Block의 SAD 연산기 구조

4. 구현 결과

본 논문에서는 SAD 연산에서 1의 보수의 수 체계 방법을 사용하여 하드웨어로 구현 했을 때 게이트 수를 줄일 수 있다는 가설을 성공적으로 입증하였다. SAD 연산기는 VHDL(Very High Speed Intergrated Circuit Hardware Description Language)를 이용하여 하드웨어로 설계하였다. 설계는 VHDL 하향식(top-down) 방식을 통해 이루어졌고, IEEE 표준 라이브러리만을 사용하였다. 설계된 SAD 연산기의 simulation을 위해 Cadence의 Simvision을 사용하여 시뮬레이션 결과를 관찰해보았다. 또한, 하드웨어를 검증하기위해 Synopsys의 Design Compiler로 합성을 수행하였다.

게이트 수는 공정 종류와 클럭 동작수에 따라 달라지므로, 공정 종류와 클럭 동작 주파수에 대한 게이트 수도 비교를 해보았다. 움직임 추정 기법에서 사용되는 여러 가지 선택사항 중에 블록 사이즈 변화에 대한 게이트 수와 탐색 범위 변화에 대한 게이트 수를 비교하는 실험도 해보았다. 실험 조건은 SAD 연산을 할 때, 가장 차이가 많이 나는 0과 255의 차를 이용하여 최대 게이트 수에 대한 결과를 살펴보았다.

1. 공정과 클럭 동작 주파수에 따른 게이트 수 비교

0.09㎛ 공정, 0.13㎛ 공정, 0.18㎛ 공정, 0.25㎛ 공정과 클럭 동작 주파수는 100Mhz, 130Mhz, 170Mhz, 200Mhz에서 블록 사이즈가 16×16일 경우에 각각의 조건에 따른 게이트 수는 표 2와 같은 결과를 나타내었다. 1의 보수의 수 체계 방법을 사용한 SAD 연산기의 게이트 수가 2의 보수의 수 체계 방법에 비해 평균적으로 12.5%의 감소율를 나타내었다.

표 2. 공정과 클럭 동작 주파수에 대한 게이트 수

공정 (㎛)	클럭 (MHz)	2's	1's	감소율(%)
0.09	100	23,681	22,938	3.1
	130	23,622	22,975	2.7
	170	23,624	23,042	2.4
	200	23,629	23,045	2.4
0.13	100	25,964	25,223	2.8
	130	26,421	25,190	4.6
	170	32,730	25,347	22.5
	200	46,999	29,180	37.9
0.18	100	25,933	25,770	0.6
	130	29,413	27,230	7.4
	170	37,382	35,894	3.9
	200	40,645	38,138	6.1
0.25	100	38,101	21,398	43.8
	130	46,995	27,046	42.4
	170	47,309	43,123	8.8
	200	48,233	43,941	8.8
Average				12.5

2. 블록 사이즈에 대한 게이트 수

H.264/AVC에서는 다양한 블록사이즈에 대한 움직임 추정을 할 수 있다. 블록 사이즈에 대한 BMA 연산기에 수도 달라진다. 표 3은 0.25㎛ 공정에서 클럭 동작 주파수 100Mhz, 블록 사이즈는 16×16, 8×8, 4×4로 실험을 한 결과를 나타내었다.

표 3. 블록 사이즈에 대한 게이트 수

블록 사이즈	클럭 (MHz)	2's	1's	감소율(%)
4×4	100	2,381	1,337	43.8
	130	2,937	1,690	42.4
	170	2,957	2,695	8.8
	200	3,015	2,746	8.9
8×8	100	9,525	5,350	43.8
	130	11,749	6,762	42.4
	170	11,827	10,781	8.8
	200	12,058	10,985	8.9
16×16	100	38,101	21,398	43.8
	130	46,995	27,046	42.4
	170	47,309	43,123	8.8
	200	48,233	43,941	8.8
Average				19.4

표 3과 같이 각 블록 사이즈마다 게이트 수의 감소를 나타나는데 평균적으로 19.4% 감소하였다.

3. 탐색 영역 크기에 대한 게이트 수

H.264/AVC에서는 다양한 탐색 영역 크기를 다루며, 움직임 추정을 수행하고 있다. 탐색 영역 크기는 연산량에 많은 영향을 끼친다. 또한, 게이트수의 증가에 많은 영향을 끼친다. 본 논문에서 16×16 블록 사이즈로 탐색 영역 크기를 정한후 구현한 H/W의 게이트 수를 표 4에 나타내었다.

표 4. 탐색 영역 사이즈에 대한 게이트 수

Size	클럭 (MHz)	2's	1's	감소율 (%)
-16,+16	100	609,616	342,368	43.8
	130	751,920	432,736	42.4
	170	756,944	689,968	8.8
	200	771,728	703,056	8.9
-32,+32	100	1,219,232	684,736	43.8
	130	1,503,840	865,472	42.4
	170	1,513,888	1,379,936	8.8
	200	1,543,456	1,406,112	8.9
-64,+64	100	2,438,464	1,369,472	43.8
	130	3,007,680	1,730,944	42.4
	170	3,027,776	2,759,872	8.8
	200	3,086,912	2,812,224	8.9
Average				25

표 4의 결과를 보면, 탐색 영역 크기에 따라 게이트 수가 상당히 많이 증가하는 것을 볼 수 있다. 이는 H.264/AVC에서 움직임 추정을 위한하드웨어의 비중을 짐작할 수 있는 구현 결과이다. 제안한 기법의 구현 결과가 이전 기법에 비해 하드웨어 자원 사용량을 줄일 수 있음을 확인 할 수 있다.

5. 결 론

본 논문에서는 H.264/AVC의 움직임 추정기의 H/W 자원 사용량을 줄이기 위한 효과적인 방법으로 SAD 연산기를 1의 보수의 연산법을 이용하는 기법을 제안하였고 이를 하드웨어로 구현하였다. 하드웨어 구현 결과 최대 하드웨어 자원 사용량을 25%이상 줄임으로써 그 효율성을 증명했다. 현재 영상의 크기가 HD급이 보통화 되고 있고, 점점 영상 사이즈가 커짐에 따라 하드웨어 구현 연구는 끊임없이 진행되리라 생각된다. 제안한 방법을 기반으로 저전력 SoC에 관한 연구를 계속 진행할 것이다.

참 고 문 헌

[1] I. Chisalita and N. Shahmehri, "Issues in image Utilization within Mobile E-Services," IEEE Workshop in Mobile Internet and E-Business Applications, Boston, USA, pp. 62-67, June 2001.

[2] A. Luthra, G.J. Sullivan, and T. Wiegand, "Introduction to the special issue in the H.264/AVC video coding standard," IEEE Trans. Circuits Syst. Video technol, vol. 13, pp. 557-559, July 2003.

[3] S. Zafar, Y-Q, Zhang, and J. S. Bars, "Predictive block - matching estimation for TV coding-Part I: Inter-block prediction," IEEE Trans. Broadcast., vol. 37, pp. 97-101, Sept. 1991.

[4] P. I Hosur and K. K. Ma, "Motion vector field adaptive fast motion estimation," Proc. Second Intern. Conf. in information, Communications and Signal Processing(ICICS'99), Singapore, Dec. 1999.

[5] A.M. Tourapis, O.C.Au, and M.L.Liou, "Predictive motion vector field search technique (PMVFAST)-Enhancing block based motion estimation," Proc. SPIE Visual Common. Image Process, San Jose, CA, Jan. 2001.