메모리 호출과 연산횟수 감소기법을 이용한 저전력 움직임추정 VLSI 구현

준회원 문 지 경*, 정회원 김 남 섭**, 종신회원 김 진 상*, 정회원 조 원 경*

VLSI Implementation of Low-Power Motion Estimation Using Reduced Memory Accesses and Computations

Ji-Kyung Moon *Associate Member*, Namsub Kim *Reguler Member*
Jinsang Kim *Lifelong Member*, Won-Kyung Cho *Reguler Member*

요 약

저전력 움직임추정은 휴대용 정보단말의 실시간 비디오 코딩에 필수적이다. 본 논문에서는 전역탐색 블럭정합 방식을 적용한 저전력 움직임추정 알고리즘과 이를 1차원 배열의 VLSI로 구현한 하드웨어 구조를 제안한다. 전역탐색 블럭정합 방법의 전력소비의 주원인은 많은 연산량과 탐색영역의 프레임 데이터를 호출하는 횟수가 많다는 점이다. 본 논문에서는 두 개의 인접한 참조블럭의 움직임추정 연산을 동시에 병렬로 수행하여 탐색영역의 메모리 호출횟수를 감소시켰으며, 움직임추정시 결과에 영향을 미치지 않는 불필요한 연산을 제거하였다. 제안된 움직임추정 알고리즘을 1차원 PE (processing element) 배열구조의 VLSI로 구현하여 실험한 결과, 제안된 움직임추정기는 기존의 저전력 움직임추정기에 비해 9.3%의 소비전력 감소와 2배 정도의 속도향상이 있음을 확인하였다.

Key Words : 전역탐색 블럭정합 방식, 저전력 구조, 움직임추정, VLSI, 시스토릭 어레이

ABSTRACT

Low-power motion estimation is required for video coding in portable information devices. In this paper, we propose a low-power motion estimation algorithm and 1-D systolic array VLSI architecture using full search block matching algorithm (FSBMA). Main power dissipation sources of FSBMA are complex computations and frequent memory accesses for data in the search area. In the proposed algorithm, memory accesses and computations are reduced by using 1D PE (processing array) array architecture performing motion estimation of two neighboring blocks in parallel and by skipping unnecessary computations during motion estimation. The VLSI implementation results of the algorithm show that the proposed VLSI architecture can save 9.3% power dissipation and can operate two times faster than an existing low-power motion estimator.

Ⅰ. 서 론

동영상 압축과정에 필수적인 움직임추정 과정은 많은 메모리 호출과 연산을 필요로 한다. 예를 들어, ITU H.261 권고의 움직임추정에 필요한 계산량은 약 1.2 GOPS (Giga Operation Per Second) 이며, 이때 소비되는 전력은 전체 동영상 압축과정의 약 50%를 차지한다.[1] 따라서 동영상 실시간 처리가 필

※ 본 연구는 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임 (과제번호: R05-2004-000-12487-0)
* 경희대학교 전자전파공학과 ({jskim27@khu.ac.kr),
** 한림대학교 전자공학과
논문번호 : KICS2004-10-218, 접수일자 : 2004년 10월 6일, 최종논문접수일자 : 2007년 4월 23일

요한 휴대 정보단말기 구현을 위하여 저전력 움직임추정 VLSI 설계가 필수적이다..

움직임추정 VLSI에서 가장 많이 사용되는 움직임추정 방식은 하드웨어 구현이 간단한 전역탐색 블럭정합 방식 (FSBMA: full search block matching algorithm) 이다.[3],[4] 그림 1과 같이 전역탐색 블럭정합 방식의 움직임벡터 추정을 위해서 이전 프레임의 탐색영역내의 모든 후보 블럭들을 탐색하여 현재 프레임의 참조블럭과 가장 잘 정합되는 이전 프레임의 블럭의 위치를 탐색하는 과정이 필요하다. 이와같은 방식은 블럭 내의 모든 화소들을 이용하여 정합하므로, 복원된 프레임의 화질이 우수하지만 연산량이 너무 많다는 단점이 있다. 이와 같은 단점을 해결하기 위하여 고속 블럭정합 방식들이 제안되었으나 고속 블록정합 방식은 블럭정합시 모든 픽셀을 이용하여 정합하지 않기 때문에 정합오류가 전역탐색 블럭정합 방식보다 커져서 복원된 화질이 떨어진다는 단점이 있다. 그러므로 전역탐색 블럭정합 방식의 고속화 및 저전력 설계를 통하여 우수한 화질을 제공할 수 있는 방식에 대한 연구는 동영상 코딩기능을 필요로 하는 휴대용 정보단말기 응용을 위하여 필수적이다 .[2],[3],[5] 전역탐색 블럭정합 방식의 많은 연산량으로 인하여 소비전력이 증가되는데 이를 감소시키기 위하여 외부 메모리 호출횟수를 감소시키는 방법 [3]을 제안하였으며, 움직임추정시 불필요한 연산을 제거시키는 방법 [7]과 이를 1D 하드웨어 배열로 설계한 연구가 있었다.[4] 또한 픽셀을 표현하는 비트수를 줄여 움직임추정 하드웨어의 복잡도를 감소시키고 처리속도를 증가시키는 구조가 제안되었으나[6] 복원된 프레임의 화질이 저하된다는 단점이 있다.

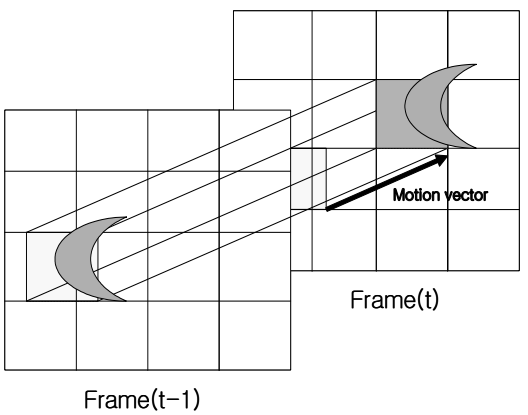


그림 1. 블럭정합방식을 이용한 움직임 예측

본 논문에서는 블럭정합시 인접블럭간의 데이터 공유를 통하여 전역탐색시 필요한 메모리 호출횟수를 감소시키며, 블럭정합시 블럭내의 두 화소간의 누적오차가 지금까지 계산된 최소 블럭오류값 보다 크면, 탐색영역내의 나머지 화소간의 연산을 중단하고 바로 다음 탐색영역의 블럭정합을 수행한다. 이와 같은 방식을 적용하면 움직임추정과정의 불필요한 연산을 줄일 수 있고, 두 블록에 대한 움직임추정을 동시에 처리할 수 있기 때문에 기존의 저전력 움직임추정 방식보다 전력소비가 적고 고속으로 처리 가능한 움직임추정을 할 수 있다.

본 논문의 구성은 다음과 같다. II 장과 III 장에서는 각각 저전력 움직임추정 알고리즘과 제안된 VLSI 구조를 기술한다. IV 장에서는 실험 결과 및 분석에 대하여 논의하며, 마지막으로 V장에서 결론을 맺는다.

Ⅱ. 전역탐색 블럭정합 방식을 이용한 저전력 움직임추정 알고리즘

전역탐색 블록정합 방식은 현재 프레임의 $N\times N$ 크기의 참조블럭 (reference block)에 대해서 이전 프레임의 탐색영역내의 모든 탐색블럭 (search blocks)과의 정합을 통하여 정합오류가 최소가 되는 탐색블럭을 찾아 이를 이용하여 움직임벡터를 추정하는 방식이다. 블럭정합은 두 블럭간의 유사성을 계산하는 과정이고, 움직임추정기 하드웨어 구현시 일반적으로 (1)식과 같이, 하드웨어 구현이 용이한 SAD (sum of absolute differences) 를 정합오류값으로 이용한다.

$$SAD(u,v)=\sum_{j=0}^{N-1}\sum_{i=0}^{N-1}|I_t(i,j)-I_{t-1}(i+u,j+v)| \quad -p\le u,v\le p \tag{1}$$

이때, $I_t(i,j)$와 $I_{t-1}(i+u,j+v)$는 각각 현재 프레임의 참조블럭과 이전 프레임의 탐색영역내의 화소값이며, p는 탐색영역의 범위이며, (u, v)는 탐색블럭의 위치이며 이 중 하나가 움직임벡터가 된다.

2.1 탐색영역의 메모리 공유를 이용한 PE 구조

FSBMA에서 $N\times N$ 블럭으로 나눠진 참조블럭들은 고정된 탐색영역을 가지게 되고 그림 2와 같이 서로 인접한 두 개의 참조블럭에 대한 탐색영역이 중첩되는 부분이 생기게 된다. 이를 이용해서 두 개의 인접한 참조블럭에 대한 블럭정합을 동시에 수

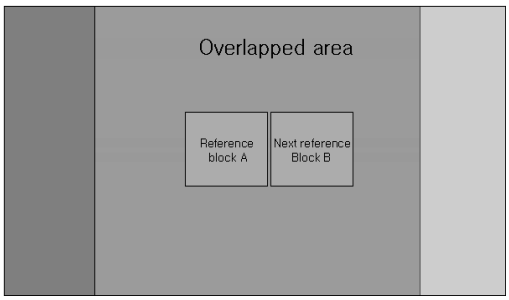


그림 2. 인접한 참조블럭의 중첩되는 탐색영역

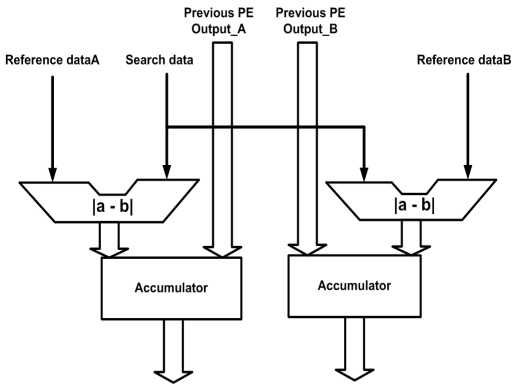


그림 3. 두 개의 SAD연산을 수행하는 PE구조

행하여 중첩되는 탐색영역내의 화소값을 인접한 두 개의 참조블럭이 공유하면 움직임추정 하드웨어 구현시 탐색영역 데이터를 저장하고 있는 메모리의 호출을 줄일 수 있으므로 소비전력을 감소시킬 수 있다는 장점이 있다.[4]

움직임추정기를 PE의 1차원 배열구조로 구현할 경우, 하나의 PE에서 두 개의 참조블럭에 대한 SAD 연산을 수행할 수 있도록 설계하면 탐색영역내의 중첩되는 데이터를 재사용할 수 있다. 그림 3은 이와 같은 기능을 수행하는 PE의 구조이며 SAD 연산을 위한 감산기와 누적기를 두개씩 사용한다. 그림 3의 구조를 1차원 PE 시스토릭 배열구조로 확장하기 위해서는 이전의 PE의 연산결과가 누적되고 두 개의 감산기에서 탐색영역의 데이터가 공유되도록 PE를 설계해야 한다.

2.2 SAD 연산에 필요한 절대차 연산횟수 감소방법

FSBMA 방식의 움직임벡터 추정과정인 (1)식의 2차원 SAD 연산을 반복적인 1차원 SAD 연산으로 표현하면 (2)식과 (3)식과 같다. (4)식과 같이 탐색영역내의 최소가 되는 SAD 값을 가진 탐색블럭의 좌표가 움직임벡터 (u,v)가 된다.

$$D_i(u,v)=\sum_{j=0}^{N-1}\left|I_t(i,j)-I_{t-1}(i+u,j+v)\right| \tag{2}$$

$$SAD(u,v)=\sum_{i=0}^{N-1}D_i(u,v) \tag{3}$$

$$(u,v)=\min_{u,v}\{SAD(u,v)\} \qquad -p\le u,v\le p \tag{4}$$

(4)식의 움직임벡터값, (u,v)를 계산하려면, 탐색영역내의 새로운 탐색블럭이 참조블럭과 정합될 때마다 탐색영역내의 이전 블럭까지의 최소 정합 오류값과 현재의 탐색영역의 블록정합 오류값을 비교해야 한다. 이때 현재 탐색영역내의 블럭정합 오류값이 이전 블럭까지의 최소 정합오류값 보다 작을 때만 최소 정합오류값을 갱신하면 된다. (3)식의 $SAD(u,v)$가 계산되는 과정을 (5)식과 같이 반복적인 식으로 표현할 수 있다. (5)식의 i 에 대하여 $D_i(u,v)$ 값이 누적되는 과정에서 $i-1$까지 누적된 결과값, $SAD_{i-1}(u,v)$ 값이 지금까지 계산된 최소 블럭정합 오류값, $SAD_{\min\_tmp}$ 보다 작을 때만 메모리에서 화소값을 호출하여 $D_i(u,v)$ 값을 갱신하고, 그 외는 이전 값, $D_{i-1}(u,v)$ 으로 대치하여도 추정된 움직임벡터는 동일함을 (6)-(8)식을 보면 쉽게 알 수 있다. 이와같이 SAD(u,v) 값 계산을 위하여 (5)식과 같이 반복적인 연산이 수행될 때마다 지금까지 계산된 최소 블럭정합 오류값과 비교하는 과정을 추가하면, 필요 없는 연산과정을 생략할 수 있어서 움직임추정 속도도 빨라지고 메모리에서 PE로 입력되는 데이터를 메모리로부터 호출할 필요가 없어져 전력소모도 감소된다.

$$SAD_i(u,v)=SAD_{i-1}(u,v)+D_i(u,v),\ 0\le i<N-1 \tag{5}$$

$$D_i(u,v)=\begin{cases}\sum_{j=0}^{N-1}\left|I_t(i,j)-I_{t-1}(u+i,c+j)\right|, & \text{if } SAD_{i-1}(u,v)<SAD_{\min\_tmp}\\ D_{i-1}(u,v), & \text{otherwise}\end{cases} \tag{6}$$

$$SAD(u,v)=SAD_{N-1}(u,v), \tag{7}$$

$$SAD_{\min\_tmp}=\min\{SAD(u,v)\} \tag{8}$$

(5)식과 같이 SAD(u,v) 연산을 반복식으로 표현하면 움직임추정 과정에 필요한 연산과정을 PE의 1차원 시스토릭 배열구조 (systolic array architecture) 로 구현하여 고속화시킬 수 있다.

Ⅲ. 제안된 저전력 움직임추정 VLSI 구조

2장에서 설명한 두 개의 참조블럭을 동시에 처리하여 탐색영역의 메모리를 공유하는 방법과 절대차 연산횟수 감소방법을 적용한 FSBMA 움직임추정

1. $SAD_{b_k,\,\min\_tmp}=\infty,\ k=1,2,\ SAD_{b_k,\,i_{k\_start}-1}(u_{k,start},\ -p)=0$
2. ***for*** $u_k=-u_{k,start}:-u_{k,start}+2p$ // *horizontal axis of search area*
3. ***for*** $v=-p:p$ // *vertical axis of search area*
4. ***for*** $i_k=i_{k,start}:i_{k,start}+N-1$ // *horizontal axis of reference area*
5. ***for*** $j=0:N-1$ // *vertical axis of reference area*
6. ***if*** $SAD_{b_k,i_k-1}(u_k,\ v)<SAD_{b_k,\,\min\_tmp}$
7. $D_{b_k,i_k}(u_k,\ v)=\sum_{j=0}^{N-1}\left|I_{b_k,t}(i_k,j)-I_{b_k,t-1}(i_k+u_k,\ j+v)\right|$
8. ***else***
9. $D_{b_k,i_k}(u_k,\ v)=D_{b_k,i_k-1}(u_k,\ v)$
10. ***end*** {*if*}
11. ***end*** {*j*}
12. $SAD_{b_k,i_k}(u_k,\ v)=SAD_{b_k,i_k-1}(u_k,\ v)+D_{b_k,i_k}(u_k,\ v)$
13. ***end*** {i_k}
14. $SAD_{b_k}(u_k,\ v)=SAD_{b_k,i_{k\_start}+N-1}(u_k,\ v)$
15. $SAD_{b_k,\,\min\_tmp}=\min\{SAD_{b_k}(u_k,\ v)\}$
16. ***end*** {*v*}
17. ***end*** {u_k}
18. $(u_k,\ v)=\arg\min_{u_k,v}\{SAD_{b_k,\,\min\_tmp}\}$ // *motion vector of block* b_k

그림 4. 제안된 저전력 움직임추정 알고리즘의 pseudo code

알고리즘의 pseudo code는 그림4와 같다.

먼저 두 개의 참조블럭 (b_1, b_2)에 관련된 내부변수들을 초기화 (1 라인) 하고, 각 참조블럭에 해당되는 탐색영역 전체 (2~3 라인) 에 대하여 $N\times N$ 블럭크기 (4~5 라인) 내에 있는 화소값에 대하여 블럭정합을 수행한다. 이때, 두 개의 참조블럭에 해당되는 탐색영역의 데이터가 중첩되므로 u_1의 시작점, $-u_{1,start}$=-p 이며 u_2의 시작점, $-u_{2,start}$=-p+N이 되어, 가로축 방향으로 탐색영역내의 데이터가 중첩됨을 알 수 있다. 두개의 참조블럭에 대한 SAD 연산 (7~14 라인)은 하나의 PE에서 수행되도록 설계하여 탐색영역의 데이터를 공유할 수 있게 한다. 참조영역과 탐색영역내의 블럭정합은 수직방향으로 먼저 수행된 후 (5 라인) 수평방향으로 수행되며, 이전의 수평방향으로 누적된 SAD 값이 임시의 최소 SAD 값보다 작을 경우에만 (6 라인) 한 블럭내에서 수직방향으로 절대차 연산의 누적 (7 라인) 이 수행되어 움직임추정시 절대차 연산횟수가 감소된다. 탐색영역의 한 블럭에 대한 정합이 수행되면 최소값을 갱신하며 (14~15 라인), 탐색영역의 모든 블럭에 대한 정합이 완료되면 각 블럭에 대한 임시의 최소 SAD 값을 가진 탐색영역내의 블럭의 좌표가 움직임벡터로 추정된다.

그림 5는 블럭의 크기가 16×16일때, 상기의 저전력 움직임추정 알고리즘을 1차원 시스토릭 배열 구조를 이용하여 설계한 VLSI 구조이다. 저전력 움직임추정기 구조는 참조블럭의 파이프라인처리를 위한 레지스터, 참조블럭과 탐색영역 데이터의 SAD 연산을 위한 16개의 PE, SAD값의 연속적인 저장을 위한 쉬프트 레지스터 (Control Shift_Reg), SAD의 최소값을 저장하긴 위한 레지스터 (D_min_Reg), 움직임벡터를 구하기 위한 카운터 (Counter), 그 외, 전체 블럭의 제어를 위한 제어블럭으로 이루어져 있다. 참조블럭의 파이프라인 처리를 위해서는 1 클럭에 참조블럭 메모리의 최대 16개의 어드레스에

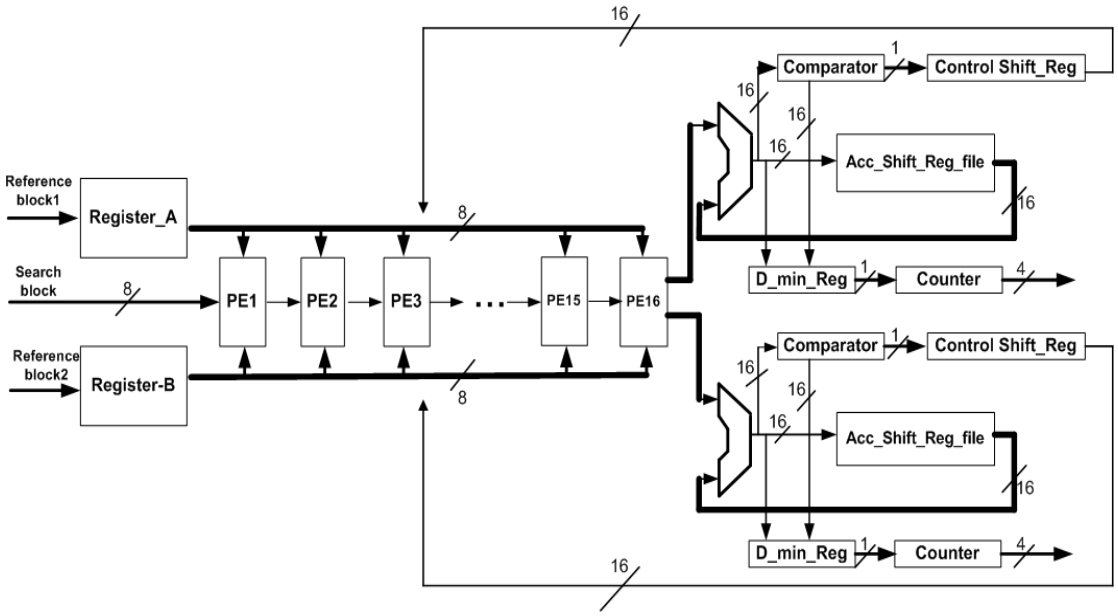


그림 5. 제안된 저전력 움직임추정기의 VLSI 구조

해당하는 데이터가 필요하게 된다. 그러므로 1클럭에 최대 16개의 동시연산을 위해서는 참조블럭의 데이터를 미리 저장하기 위한 레지스터가 필요하게 된다 (Register_A, Register_B). 16개의 PE들은 그림 3과 같이 두 개의 SAD 연산을 동시에 처리하는 구조로 이루어져 있다. PE에 참조블럭의 데이터와 탐색영역의 데이터가 입력되어, 두 데이터의 차의 절대값과 이전 PE에서 들어오는 값을 더해서 저장하게 된다.

처음 16번째 클럭에는 첫 번째 참조블럭의 데이터만 입력되고 17번째 클럭부터 첫 번째 참조블럭의 첫 연산결과가 출력되며, 이때 두 번째 참조블럭의 데이터가 입력된다. 33번째 클럭에 두 번째 참조블럭의 첫 연산 결과가 출력되며, 722번째 클럭에 첫 번째 참조블럭의 움직임벡터 추정결과가 출력되며, 738번째 클럭에서 두 번째 참조블럭의 결과가 출력된다.

PE의 연산 결과가 한 횡씩 출력되기 때문에 Acc_Shift_Reg_file을 이용하여 각 횡의 중간 값을 저장한다. D_min_Reg는 SAD 연산결과의 최소값을 저장하고 Control_Shift_Reg는 각 PE 들의 동작을 제어한다. Control_shift_Reg는 현재 누적되고 있는 SAD연산 값이 D_min_reg에 저장되어 있는 값보다 클 경우 더 이상 SAD 연산을 수행하지 않기 위하여 Comparator의 출력을 이용하여 16비트 제어신호를 각 PE에 전달시킨다. 그 결과 불필요한 연산을 수행하지 않아 움직임추정 속도가 빨라지고 탐색영역의 메모리에서 PE로 입력되는 데이터를 호출할 필요가 없어져 전력소모가 감소된다. 카운터는 움직임벡터의 출력을 위한 것으로 클럭을 카운트해서 D_min_reg에 저장되어 있는 값이 최소일 경우에 움직임벡터를 생성한다.

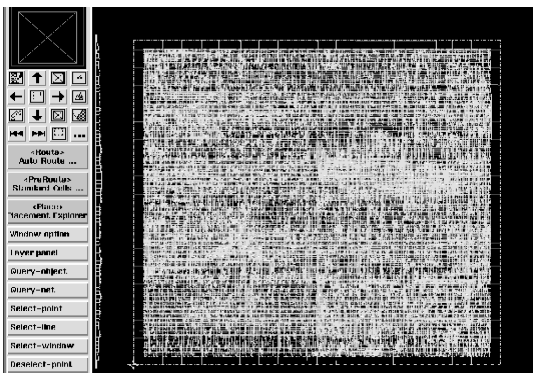


그림 6. 제안된 움직임추정기의 레이아웃 결과

Ⅳ. 실험결과 및 분석

제안된 저전력 움직임추정기와 탐색영역의 절대차 연산횟수 감소방법만을 적용한 기존의 1D 시스토릭 배열구조의 움직임추정기 [4]의 성능을 비교분석하였다. 우선, VHDL 언어를 사용하여 제안된 움직임추정기 (Sim3의 파라미터를 이용) 와 기존의 움직임추정기를 모델링하고 기능검증을 거쳐 Synopsys Design Analyzer를 사용하여 합성을 한 후, 네트리스트 (netlist)를 추출하였다. 게이트 수준의 검증 후, 추출된 네트리스트를 가지고 Apollo II 툴을 사용하여 P&R (Place & Route) 을 수행하였다. 그 그림 6은 제안된 움직임추정기의 P&R 결과를 나타낸 것이다. P&R이 끝나면 SDF (Standard Delay Format), SPEF (Standard Parastic Extraction Format) 파일을 추출하고 이것을 이용하여 SAIF (Switching Activity Interchange Format) 를 생성하고 Synopsys사의 Power Compiler 도구로 소비전력, 면적, 속도를 측정하였다. 본 실험에서는 UMC 0.25um 공정 라이브러리를 사용하였다.

소비전력 측정에 사용된 동영상 (miss_america, table_tennis, flower_garden) 의 블럭 크기는 16×16 이고 탐색 범위는 -8~ +8이다. 표 1은 제안된 움직임추정기와 기존의 움직임추정기의 소모 전력을 비교한 결과이다. Miss America 동영상의 소비전력은 기존의 움직임추정기에 비해 8.92mW가 감소하였고, Table Tennis의 경우에는 9.35mW가 감소하였으며, Flower Garden의 경우에는 7.45mW가 감소하였다. Flower Garden 동영상의 전력감소값이 제일 작은 이유는 Flower Garden 동영상이 고주파 성분이 많기 때문이다. 제안된 움직임추정기의 소모전력은 기존의 움직임추정기에 비해 평균적으로 9.3% 정도 감소하는 것을 알 수 있다.

표 2는 제안된 움직임추정기와 기존의 움직임추정기의 처리속도와 면적을 분석한 결과이다. 제안된 움직임추정기는 두 개의 인접한 참조블럭을 동시에 처리하기 위한 블럭이 추가적으로 필요하여, 기존의 움직임추정기에 비해 1.9배의 면적이 증가하게 된다. 제안된 움직임추정기는 기존의 움직임추정기보다 면적이 증가하여 이에 상응하는 소비전력의 증가가 두 블럭의 절대차 연산횟수의 감소로 인한 소비전력 감소보다 작아 전체 소비전력이 감소됨을 알 수 있다. 그러나 제안된 구조는 두 개의 인접한 참조블럭을 동시에 수행하기 때문에 기존의 움직임추정기에 비해 2배정도의 빠른 연산 속도를 갖음을 알 수 있다.

표 1. 움직임추정기의 소모전력 분석(단위 : mW)

	기존의 움직임 추정기	제안된 움직임 추정기
Miss America	91.27	82.35
Table Tennis	94.66	85.31
Flower Garden	98.12	90.67

표 2. 움직임추정기의 처리속도와 면적 비교

	기존의 움직임 추정기	제안된 움직임 추정기
계산 시간 (ms)	5.0	2.5
면적 (CLBs)	206495	393788

V. 결 론

본 논문에서는 휴대용 멀티미디어 정보단말에 필수적인 저전력 움직임추정기를 제안하였다. 인접 블럭간의 메모리 공유와 절대차 연산횟수 감소방법을 적용하여 1차원 PE 배열 구조에 간단한 부가적인 하드웨어를 추가하여 이를 VLSI로 구현하였다. 인접한 두 개의 참조블럭이 호출하는 탐색영역의 데이터를 공유해서 사용함으로써 메모리의 데이터를 호출하는데 소모되는 전력을 감소시켰으며 PE의 SAD연산의 최소값을 레지스터에 저장하여 순차적으로 들어오는 SAD값과 비교하여 새로운 연산결과값이 레지스터에 저장되어 있는 값보다 클 경우에는 PE의 연산을 더 이상 수행되지 않게 하여 연산속도를 증가시켰다. VLSI 구현실험을 통한 성능분석 결과, 제안된 움직임추정기는 기존의 움직임추정기에 비해 약 2배정도의 면적이 증가하였으나, QCIF 동영상에 대하여 평균 9.3%의 소비전력 감소가 있었고 기존의 움직임추정기보다 2배 정도의 고속 움직임추정이 가능함을 확인하였다.

참 고 문 헌

〔1〕 K. Guttag et al. "a single-Chip Multiprocessor For Multimedia: The MVP," IEEE Computer Graphics and Applications, Nov, 1992. pp 53-64.

〔2〕 Elgamel, M.A.; Shams, A.M.; Bayoumi, M.A.; "A comparative analysis for low power motion estimation VLSI architectures," 2000 IEEE Workshop on SiPS, 2000 Page(s): 149-158.

〔3〕 Bo-Sung Kim; Jun-Dong Cho, "VLSI architecture for low power motion estimation using high data access reuse," ASICs, 1999. AP-ASIC '99. The First IEEE Asia Pacific Conference on 1999, Page(s): 162-165.

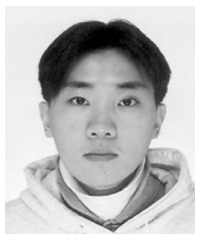
〔4〕 L. Sousa, and N. Roma, "Low-Power Array Architectures for motion Estimation," Proceeding of the IEEE international Workshop on Multimedia Signal Processing, Copenhagen, MMSP'99, pp. 679-684, Copenhagen, Denmark, September 1999. Co-Chaired by Ed Deprettere and Bastiaan Kleijm.

〔5〕 C. Hsieh, and T. Lin, "VLSI Architecture for Block-Matching Motion Estimation Algorithm," IEEE Transactions on Circuits and Systems for Video Technology, vol. 2, pp. 169-175, 1992.

〔6〕 Z. He, and M. Liou, "Reduing Hardware Complexity of Motion Estimation Algorithms Using Truncated Pixels," IEEE International Symposium on Circuits and Systems 1997, ISCAS'97, pp. 2809-2817, Hong Kong, June 1997.

〔7〕 V. Do, and K. Yun, "A Low-Power Architecture for Full-Search Block-Matching Motion Estimation," IEEE Transactions on Circuits and Systems for Video Technology, vol. 8, pp. 393-398, 1998.

문 지 경 (Ji-Kyung Moon) 준회원



2003년 2월 : 강남대학교 전자공학과 학사

2005년 2월 : 경희대학교 전자공학과 석사

<관심분야> 영상처리, VLSI 시스템 설계

김 남 섭 (Namsub Kim) 정회원

1990년 경희대학교 전자공학과 학사

1992년 경희대학교 전자공학과 석사

2006년 경희대학교 전자공학과 박사

2007년~한림대학교 전자공학과 강의전임교수

<관심분야> SoC 설계 및 테스트, 멀티미디어 ASIC, Computer Vision, Ubiquitous Health Care

김 진 상 (Jinsang Kim) 종신회원

2000년 12월 : 미국 콜로라도 주립대 전자공학박사
1990년 2월~2001년 8월 : KT 연구소
2001년~현재 : 경희대학교 전자정보학부 조교수
<관심분야> 영상처리와 이동통신용 SoC 설계

조 원 경 (Won-Kyung Cho) 정회원

1986년 8월 : 한양대학교 전자공학과 공학박사
1980년~현재 : 경희대학교 전자정보학부 정교수
<관심분야> 컴퓨터시스템 구조, VLSI 설계