에너지 관리 알고리즘을 이용한 저전력 움직임 추정기 구조

학생회원 김 응 섭*, 정회원 이 찬 호**

Low Energy Motion Estimation Architecture using Energy Management Algorithm

Eung-sup Kim* *Student Member*, Chanho Lee** *Regular Member*

요 약

모바일 기술과 개인 휴대통신 서비스가 발달함에 따라, 휴대용기기에서 멀티미디어 데이터의 연산량은 점점 증가하고 있다. 따라서 배터리로부터 에너지를 얻는 모바일 기기에서의 에너지 관리는 더욱더 중요해지고 있다. 비디오 인코딩에 필수적으로 사용되는 움직임 추정기는 매우 많은 연산량을 갖고 있어, 비디오 인코더에서 가장 많은 에너지를 소모하고 있다. 본 논문에서는 여러 가지 고속 탐색 알고리즘과 에너지 관리 알고리즘을 적용할 수 있는 저전력 움직임 추정기 구조를 제안한다. ECVH (Energy-constrained Vdd hopping)는 slack time과 주어진 에너지 여유분에 따라 수행시간에 사용하는 알고리즘과 동작 주파수, 공급 전압을 동적으로 바꾸어 에너지 소모를 줄이고 주어진 조건 내에서 성능을 최대화하는 알고리즘이다. 본 논문에서는 ECVH를 제안한 구조에 적용했을 때 움직임 추정기의 에너지 소모가 효과적으로 줄어드는 것을 시뮬레이션을 통해 보여주고 있다. 또한 ECVH 알고리즘을 수행하는 전원관리부와 움직임 추정기의 설계 결과를 보여주고 있다.

Key Words : Motion estimation, low power, ECVH, MPEG

ABSTRACT

Computation of multimedia data increases in portable devices with the advances of the mobile and personal communication services. The energy management of such devices is very important for the battery-powered operation hours. The motion estimation in a video encoder requires huge amount of computation, and hence, consumes the largest portion of the energy consumption. In this paper, we propose a novel architecture that a low energy management scheme can be applied with several fast-search algorithms. The energy-constrained Vdd hopping (ECVH) technique reduces power consumption of the motion estimation by adaptively changing the search algorithm, the operating frequency, and the supply voltage using the remaining slack time within given power-budget. We show that the ECVH can be applied to the architecture, and that the power consumption can be efficiently reduced.

Ⅰ. 서 론

이동형 기기에서의 동영상 처리에 대한 요구는 날로 증대되어, 동영상을 촬영할 수 있는 휴대폰은 보편화된 지 오래되었다. 동영상 처리는 연산량이 많아 연산 수행에 많은 에너지를 소모하게 되지만,

* 숭실대학교 전자공학과(freezer@ssu.ac.kr), ** 숭실대학교 전자공학과(chlee@ssu.ac.kr)
논문번호 : KICS2005-01-007, 접수일자 : 2005년 1월 4일
※ 본 연구는 숭실대학교 교내연구비 지원으로 수행되었습니다.

이동형 기기의 에너지 사정은 그렇게 좋지 못하다. 따라서 동영상 처리 기기에서 동영상 처리를 구현하기 위해서는 적은 에너지를 소모하게 하는 것이 가장 큰 관심사이다. 동영상 처리 중에서도 인코딩(video encoding) 부분이 에너지를 많이 소모하고 특히, 동영상 압축에 필수적으로 사용되는 움직임 추정(motion estimation) 부분이, 엄청난 연산량으로 인해 비디오 인코더(video encoder)가 소모하는 에너지의 대부분을 차지하고 있다[1]. 따라서 움직임 추정기가 소모하는 에너지를 줄이는 것이 동영상 처리에 필요한 에너지를 줄이는 데 무엇보다 효과적이라고 할 수 있다.

움직임 추정의 연산량을 줄이고자 3 step search (TSS)[2], 4 step search (FSS)와[3] 같은 여러 가지 고속 탐색 알고리즘(fast-search algorithm)이 제안되어 왔다. 하지만 일반적으로 연산량이 적은 알고리즘일수록 검색 능력은 떨어진다. 따라서 필요한 검색능력과 에너지 소모량을 만족시킬 수 있는 알고리즘을 찾기는 쉽지 않으며, 에너지와 성능에 대한 요구가 변하는 응용에 적합한 알고리즘을 찾기는 특히 어렵다.

ECVH (Energy-constrained Vdd hopping)는[4],[5] 소프트웨어 태스크 스케줄링(software task scheduling)방법의 하나로, 태스크의 완료 시간이 대부분 WCET(*the worst case execution time*)를 넘지 않는다는 점을 이용한다. 현재 태스크가 WCET에 못 미쳐 끝나게 되면 WCET까지의 남는 시간, 즉, slack time이 남게 된다. 이 slack time은 다음에 수행될 태스크에 할당이 된다. 또한, 태스크마다 일정한 에너지를 할당하여 (energy budget) 현재 태스크가 소모하고 남은 에너지는 다음에 수행될 태스크에 넘긴다. 따라서 다음 태스크는 사용 가능 시간과 에너지 할당량이 늘어나게 되고, 이렇게 늘어난 시간과 에너지 할당량을 넘지 않는 범위 내에서 가장 성능이 높은 알고리즘과 가장 적은 에너지를 소모하는 동작 주파수와 공급 전압을 선택하게 된다. 따라서 ECVH를 움직임 추정기에 적용시키게 되면, 주어진 시간과 에너지 조건을 넘지 않는 범위에서 가장 성능이 좋은 알고리즘을 수행시킬 수 있으며, 에너지 조건을 변경시킴으로써 움직임 추정기의 검색 성능과 에너지 소모를 쉽게 제어할 수 있다.

ECVH를 움직임 추정기에 적용시키기 위해서는, 여러 가지 움직임 추정 알고리즘을 수행시킬 수 있어야 하고, slack time을 만들기 위해 입력 값에 따라 되도록 빨리 연산을 끝마쳐야 한다. 이것은 소프트웨어로는 구현하기 쉽지만 하드웨어로는 구현하기 쉽지 않다. 왜냐하면, 하드웨어의 수행시간은 대부분 일정하기 때문에 ECVH를 적용시킬 만한 slack time을 얻기가 쉽지 않기 때문이다.

본 논문에서는, ECVH를 적용할 수 있도록 입력 값에 따라 되도록 빨리 연산을 끝내며, 여러 가지 움직임 추정 알고리즘을 수행할 수 있는 움직임 추정기 구조를 제안한다. 제안하는 구조는 H. Jong이 제안한 TSS 알고리즘을 수행할 수 있는 움직임 추정기 구조를 기초로 삼는다[6]. Jong의 구조는 수행시간이 일정하여 ECVH를 적용시킬 만한 slack time을 얻을 수 없고, 3개의 PE(processing element)를 갖는 변형 구조는 과도하게 외부 메모리를 많이 참조한다. 따라서 본 논문에서는 Jong의 구조의 메모리 모듈 구성과 연산 순서를 바꾸었으며, 여기에 더해 R. Richmond가 제안한 저전력 방법을 적용하였다[7]. 또한, 본 논문에서는 전력 관리부(PMU: power management unit)를 설계하여 ECVH를 효율적으로 수행할 수 있도록 하였다. 마지막으로, 제안한 움직임 추정기의 검색 성능과 에너지 소모를 시뮬레이션을 통해 알아보고, 하드웨어로 구현하였다.

Ⅱ. 움직임 추정기 구조

2.1 여러 가지 검색 알고리즘의 적용과 연산순서

ECVH를 적용하려면 움직임 추정기가 여러 가지 검색 알고리즘을 수행할 수 있어야 한다. 본 논문에서는 TSS[2], TSS의 multi-candidate 변형 알고리즘(MC-TSS)[8], full search (FS) 알고리즘을[9] 움직임 추정 알고리즘으로 사용하였다. FSS 알고리즘도 고려하였으나 동영상 샘플에 따라 TSS와 성능면에서 우열을 가릴 수 없고 WCET 값이 TSS의 경우보다 더 커서 현실적으로 사용이 어려워 채택하지 않았다. 하나의 PE는 매 순간, 하나의 검색지점에 대한 SAD(sum of absolute distortion)를 계산하게 되며, TSS의 경우 스텝 당 9개의 검색지점에 대한 SAD를 구하게 된다. FS 알고리즘의 경우 검색 영역 내의 모든 검색지점에 대한 SAD를 구하게 되는데, TSS의 경우처럼 검색지점을 9개(3x3)의 그룹으로 묶으면 하나의 그룹을 하나의 스텝으로 볼 수 있다. 이렇게 함으로써 FS와 TSS의 연산이 비슷해져 여러 가지 알고리즘을 지원하기 위한 하드웨어의 부담을 줄일 수 있다.

그림 1은 한 스텝의 연산 순서를 보여주고 있다. 9개의 검색지점은 3개의 행으로 나눠지고, 3개의

PE가 첫 번째 행에 대한 SAD 연산을 수행하고, 순차적으로 두 번째 행, 세 번째 행에 대한 SAD 연산을 진행하게 된다.

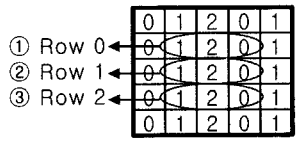


그림 1. 검색 알고리즘에서의 연산 순서

2.2 저전력 기법(low power scheme)

2.2.1 픽셀 재사용(Pixel reuse)

움직임 추정 연산 시에, 검색 영역에 위치한 픽셀(search region pixel)들은 빈번하게 참조된다. 하지만 픽셀(pixel)들을 가져오기 위해 외부 메모리를 참조하는 데는 많은 에너지가 소모되기 때문에 되도록 줄이는 것이 좋다. 그림 2를 보면 인접한 매크로 블록(macro-block)의 검색 영역(search region)이 서로 겹치는 것을 볼 수 있는데, 이러한 특성을 이용해 픽셀을 재사용함으로써 외부 메모리 참조를 줄일 수 있다. 본 논문에서는 HSA(half search area) 버퍼를 이용하여 가로로 인접한 매크로 블록 사이의 겹치는 검색영역픽셀을 재사용하는 방법을 사용하였을 뿐만 아니라[6], 세로로 인접한 매크로 블록에 대한 연산을 동시에 진행함으로써 세로로 인접한 검색영역픽셀들도 재사용하였다.

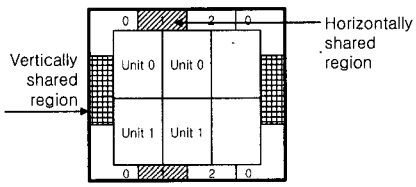


그림 2. 픽셀 재사용 (pixel reuse)

2.2.2 불필요한 연산 제거

한 행에 대한 연산이 끝나고 나면 그 행에 위치하는 3개의 검색지점 중에서 최소의 SAD를 갖는 지점을 알 수 있다. 다음 행을 계산할 때 어느 한 지점에 대한 SAD 값이 이전 행의 최소 SAD 값을 넘게 되면 그 지점에 대한 SAD를 구할 필요가 없다. 또한 TSS의 경우를 예로 들어, 2번째와 3번째 스텝에서는 이전 스텝에서 중심점에 대한 SAD를 이미 구했기 때문에 중심점의 SAD를 다시 구할 필요가 없다. 그림 3(a)는 두 번째 행의 오른쪽 지점이 이전 행의 최소 SAD 값을 넘어서 계산이 정지되는 예를 보여주고 있다.

2.2.3 이른 종료(Early termination)

하나의 행을 계산할 때, 그 행에 위치한 3개의 검색지점에 대한 SAD 값이 이전 스텝 또는 이전 행에서 계산한 최소 SAD를 넘어서 모든 PE의 동작이 중지되면, 다음 행 또는 다음 스텝으로 연산을 진행시켜 수행 사이클 수를 줄일 수 있다. 이렇게 이른 종료로 인해 남은 slack time을 이용하면, ECVH를 적용시킬 수 있다. 그림 3(b)는 첫 번째 행의 모든 검색 지점이 이전 최소 SAD 값을 넘어 다음 행으로 연산을 진행시키는 예를 보여주고 있다.

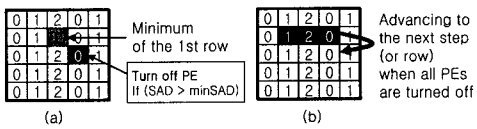


그림 3. SAD 연산 시 사용하는 저전력 기법
(a) 불필요한 연산 제거
(b) 이른 종료 (early termination)

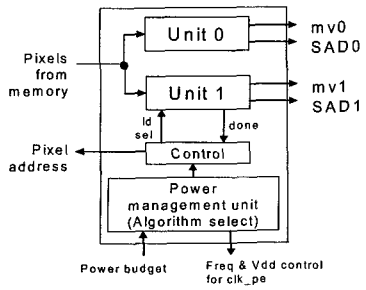
2.3 움직임 추정기 구조

2.3.1 최상위 블록(Top-level block)

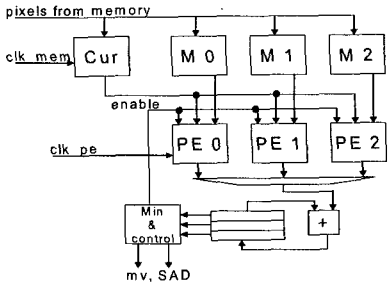
그림 4(a)는 전체 하드웨어 구조(hardware architecture)를 나타내고 있으며 최상위 블록은 연산 유닛(operation unit)들과 전원관리부(PMU), 메인 컨트롤러(main controller)로 구성된다. 그림 4(b)는 연산 유닛의 내부 구조를 보여주는 그림이다. 연산유닛은 한 번에 하나의 매크로 블록에 대한 움직임 벡터(motion vector)를 계산하며 높은 처리율을 요구하는 경우에는 연산 유닛 여러 개를 사용하여 동시에 여러 매크로 블록을 처리하게 할 수 있다. 전원관리부는 ECVH를 수행하는 블록으로 수행 시간 중에 동적으로 움직임 추정에 사용할 알고리즘 및 동작 주파수, 공급 전압을 결정하는 역할을 한다. 메인 컨트롤러는 외부 메모리 참조 및 연산 유닛을 제어한다.

2.3.2 연산 유닛(Operation unit)

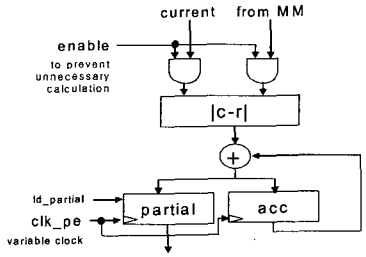
연산 유닛은 현재 프레임 픽셀 메모리(current frame pixel memory), 3개의 메모리 모듈(memory module), 그리고 3개의 PE와 누산기-큐(accumulator-queue)로 구성되어 있다. 하나의 현재 프레임 픽셀과 그에 대응되는 3개의 검색 픽셀(search point pixel)과의 차이를 3개의 PE가 구하여 누적하게 된다. Jong의 구조에서 사용한 것처럼, 3개의 검색 픽셀을 한 번에 메모리에서 읽기 위해 (3개의 port를 얻기 위해) 인터리빙(interleaving) 기법을 사용한 분산된 메모리 구조를 사용하고 있으며, 3개의 PE와



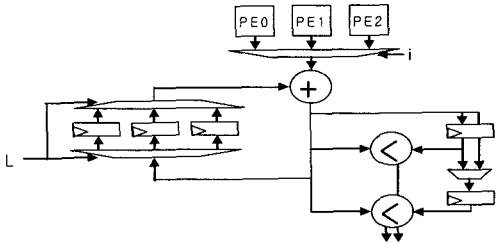
(a) 최상위 블록 (top-level block)



(b) 연산 유닛 (operation unit)



(c) PE (processing element)



(b) 누산기-큐 (accumulator-Queue)

그림 4. 제안한 움직임 추정기 구조

각 메모리 모듈들을 연결하는 연결선 수를 줄이기 위해 PE 기능 재분배(function redistribution) 방법을 사용하였다.

2.3.3 메모리 모듈(Memory module)

본 논문에서는 분산 구성된 메모리 모듈을 사용한다[6]. 단, 메모리 모듈의 수는 9개에서 3개로 통합하고 인터리빙 방법을 변경하였다. 메모리 모듈(Mi, i={0, 1, 2}로 표기)은 PE의 개수와 같이 3개가 존재하며, 하나의 메모리 모듈은 3개의 부블록(sub- block)으로 구성된다. 하나의 HSA는 메모리 모듈 당 하나씩, 3개의 부블록으로 구성된다. 이때 하나의 부블록은 하나의 포트를 갖는 메모리이다.

HSAi를 i번째(i=0, 1, 2) HSA라고 하고 HSAi가 참조 프레임 픽셀(reference frame pixel) 저장(load) 용으로 선택됐다고 가정하자. HSAi를 구성하는 검색 영역 픽셀(search region pixel)을 외부 메모리로부터 가져왔을 때, 그 픽셀을 Mi부터 시작하여 i%3, (i+1)%3, (i+2)%3의 순서대로 인터리브(interleave)한다. 이때 %는 나머지 연산자(modular operator)를 의미한다. 인터리브된 화소는 각 메모리 모듈의 i번째 부블록에 저장된다. 인접한 매크로 블록 간의 검색 영역(search region)은 한 픽셀의 간격(gap)이 존재하기 때문에 메모리 모듈 내에서 픽셀의 이동 없이 이전 검색 영역의 절반을 재사용하려면 검색 영역의 가로 폭보다 한 픽셀이 많게 HSA의 폭을 잡아야 한다[6]. 따라서 HSA의 폭은 3의 배수가 아니며, 나머지로 1이 남게 된다. 따라서, HSA의 번호와 같은 번호의 메모리 모듈부터 인터리브 해야 화소가 계속 M0, M1, M2 순서대로 저장된다는 것을 보장할 수 있다.

2.3.4 PE(Processing element)

그림 4(c)는 PE의 구조를 보여주고 있는 그림으로, Jong의 구조의 PE에 인에이블(enable) 신호를 추가하였고[7], 부분합 레지스터(partial sum register)의 위치를 바꾼 것을 알 수 있다. PE는 인에이블(enable) 신호가 활성화(active)되어 있을 때만 연산을 진행하며, 활성화되어 있지 않을 때는 입력이 뺄셈기로 들어가지 못하도록 하여 불필요한 에너지 소모를 막는다.

현재 프레임 픽셀의 비교 순서는 Jong의 구조를 따르지만 인터리브 방법이 다르기 때문에 PE 기능 재분배 순서는 달라진다. 각 행의 연산 위치를 왼쪽부터 0, 1, 2 라고 할 때, 검색 지점을 L이라고 하고, 중앙점의 픽셀이 위치한 메모리 모듈의 번호를 K(=0, 1, 2), 연산 위치간의 간격을 S, 각 매크로 블록의 행에서 PE 기능 재분배가 일어난 회수를 D라 하면, 매 순간 j번째 PEj는 j=(D+K+2S+L·S)%3의 검색지점에 대한 SAD를 계산한다.

2.3.5 누산기-큐(Accumulator-Queue)

그림 4(d)는 누산기-큐 구조를 보여주고 있다. PE 기능 재분배가 일어날 때마다 PE가 계산하는 위치가 바뀌기 때문에 그때까지 PE가 누적한 부분합은 누산기-큐에 검색 위치별로 누적해야 한다. PE 기능 재분배가 일어날 때 부분합은 그림 4(c)에서 보여주고 있는 것처럼 부분합 레지스터에 저장되게 되고 누산기-큐에 한 클럭에 하나씩 누적된다. 그 동안 PE는 새로운 위치에 대한 또 다른 부분합을 계산하게 된다. 하나의 행(row)에 대한 연산이 끝나고 나면 이전 행 또는 스텝에서 구한 최소 SAD와 현재 계산한 행에 위치한 3개의 검색 위치에 대한 SAD와 비교하여 최소 SAD를 갱신하게 된다. Multi-candidate TSS를 지원을 위해서는 최소값 다음으로 작은 SAD도 구해야 하므로, multi-candidate 수만큼의 비교기가 존재한다.

Ⅲ. ECVH의 적용

3.1 검색 알고리즘과 에너지 레벨의 결정

ECVH에서는 아래 수식에서 집합 R의 왼쪽 원소부터 차례대로 검색하여 시간 조건과 에너지 조건을 만족시키는 원소(알고리즘 및 레벨)를 선택을 하는 것이 최적이다.

Algorithm_set = {algo0, algo1, ⋯, algom-1}
where WCET(algoi) < WCET(algoj) for i<j
Energy_level_set = {level0, level1, ⋯, leveln-1}
where leveli < levelj for i<j
R = Algorithm_set x Energy_level_set
= { (algom-1, level0), (algom-1, level1),⋯,
(algo0, level0), ⋯, (algo0, leveln-1)}

그러나 위의 방법의 비교회수는 m x n에 비례하는 비교회수를 갖는다. 여기서 m은 알고리즘 수이고 n은 에너지 레벨의 수이다. 이는 과도한 비교회수이기 때문에, 본 논문에서는 먼저 아래 수식에서 집합 R'의 왼쪽 원소부터 순서대로 조건을 평가하여 시간과 에너지 조건을 만족시키는 알고리즘을 선택한 다음, 집합 L의 원소를 왼쪽부터 평가하여 에너지 레벨을 결정하는 것으로 변경하였다.

R´ = Algorithm_set x {leveln-1}
= { (algom-1, leveln-1), (algom-2, leveln-1),⋯,
(algo1, leveln-1), ⋯, (algo0, leveln-1)}
L = {algosel} x Energy_level_set
= { (algosel, leveln-1), ⋯, (algosel, level0)}

위의 식에서 $algo_{sel}$는 선택된 알고리즘이다. 이렇게 변경하면, 비교회수는 m+n에 비례하게 되어 비교회수를 줄일 수 있다. 위에서 설명한 두 가지 방법을 각각 사용했을 때, 선택된 알고리즘, 동작 주파수, 공급 전압의 차이는 매우 작았다. 따라서 비교회수가 작은 두 번째 방법을 사용한다.

3.2 전원관리부(PMU: Power management unit)

그림 5(a)에 나타나있는 전원 관리부(PMU: powr management unit)은 앞서 설명한 알고리즘에 따라 검색 알고리즘, 동작 주파수, 공급 전압을 제어한다. 전원관리부는 ECVH 유닛(ECVH unit), 주파수 및 공급 전압 선택 유닛, 캘리브레이션 유닛(calibration unit)으로 구성된다.

3.2.1 ECVH 유닛

그림 5(b)에 표시되어 있는 ECVH 유닛은 ECVH 알고리즘에 따라, 주어진 시간 및 에너지 조건에 맞는 검색 알고리즘, 동작 주파수, 공급 전압을 결정하는 역할을 한다. ECVH 유닛은 기준 클럭 주기를 하나의 단위(unit)로 삼아 시간을 카운트한다. 각 매크로 블록을 시작할 때 가장 연산량이 많은 알고리즘인 FS를 가장 빠른 클럭으로 수행시킬 때의 시간을 카운터에 더한 후, 동작 주파수의 주기와 기준 클럭 주기의 비를 매 클럭마다 감소시킨다. 하나의 매크로 블록의 연산이 끝나게 되면 남은 시간이 카운터에 기록된다. 에너지도 같은 방법으로 카운트하면 연산이 끝났을 때 에너지 할당량(energy budget)에서 쓰고 남은 에너지가 카운터에 기록된다.

남은 시간과 남은 에너지는 다음 매크로 블록에 이월되어 다음 매크로 블록이 사용할 수 있는 시간과 에너지를 늘린다. 이렇게 증가된 할당량에 대해 다음 매크로 블록에서 시간과 에너지 조건을 만족시키는 알고리즘과 에너지 레벨을 구한다. 단, 에너지 조건을 만족시킬 수 없는 경우에는 WCET가 가장 작은 알고리즘과 시간 조건을 만족시키는 가장 낮은 에너지 레벨을 선택한다. 이렇게 선택된 알고리즘은 움직임 추정기로, 동작 주파수와 공급 전압은 아래에 설명할 주파수 및 공급 전압 선택 유닛에 전달된다.

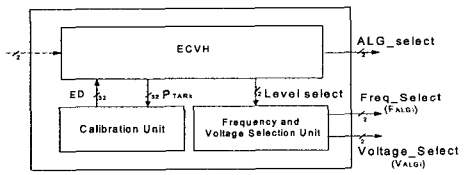
3.2.2 주파수 및 공급 전압 선택 유닛

주파수 및 공급 전압 선택 유닛(FVSU: frequency and voltage selection unit)은 ECVH 유닛에서 선

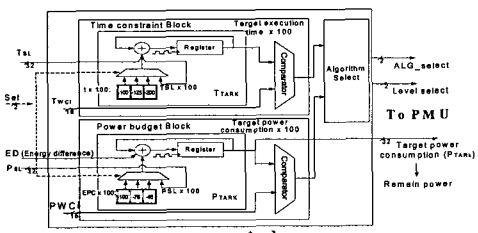
택한 에너지 레벨에 따라 다음 매크로블록 연산에서 사용할 동작 주파수와 공급 전압을 선택한다. 그림 5(c)는 FVSU의 블록도를 보여주고 있다. 또한 FVSU는 움직임 추정기 뿐만 아니라 다른 부시스템(sub-system)의 동작 주파수와 공급 전압을 선택하도록 멀티 포트(multi-port)를 제공하게 할 수 있다.

3.2.3 캘리브레이션 유닛

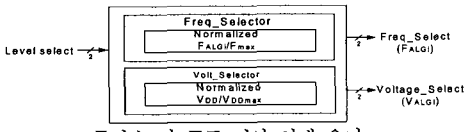
ECVH 유닛은 한 클럭 당 소모하는 에너지의 양이 일정하다고 가정하고 연산 수행 사이클 수를 셈으로써 연산에 소모된 에너지양을 추정하는 방법을 사용하였다. 그러나 실제로 한 클럭 당 소모하는 에너지의 양은 같지 않다. 따라서 실제 소모한 에너지양과 ECVH가 추정하는 에너지 소모량과의 차이를 보정해주어야 할 필요가 있다. 본 논문에서는 ECVH가 추정하는 에너지 소모량과 배터리 정보와의 차이를 보정해주는 캘리브레이션 유닛을 PMU에 추가하였다. 또한 캘리브레이션 유닛은 사용자의 필요에 따라 켜고 끌 수 있도록 하였다. 그림 5(d)는 캘리브레이션 유닛의 블록도를 보여주고 있다.



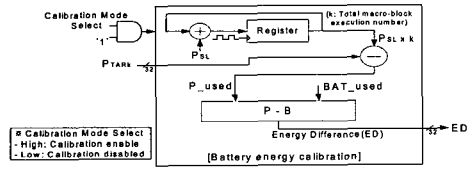
(a) PMU (Power management unit)



(b) ECVH 유닛



(c) 주파수 및 공급 전압 선택 유닛



(d) Calibration unit

그림 5. PMU (Power managment unit)

Ⅳ. 설계 및 분석 결과

본 논문에서는 제안한 움직임 추정기 구조의 특성을 시뮬레이션을 통해 분석하였다. 그림 6은 제작한 시뮬레이터의 이미지를 보여주고 있다. 이 시뮬레이터는 제안한 하드웨어 구조의 사이클 정확도 모델(cycle-accurate model)을 구현하고 있으며, 연산을 수행한 뒤에 각종 분석 자료 및 통계자료를 계산하여 보여준다. 따라서 본 시뮬레이터를 갖고 제안한 움직임 추정기 구조의 특성을 알아볼 수 있었으며, 아래에 그 결과를 나타내고 있다. 아래 자료에서 에너지 값은 가장 높은 주파수와 가장 높은 공급 전압으로 동작할 때, 한 클럭 당 소모하는 에너지를 단위로 하여 계산된 결과이다. 또한 연산 수행 시간은 가장 높은 주파수의 클럭 주기를 단위로 하여 계산된 값이다. PSNR(peak signal to noise ration)은 움직임 보정(motion compensation) 이미지와 원본 현재 프레임(current frame)을 이용하여 계산하였다.

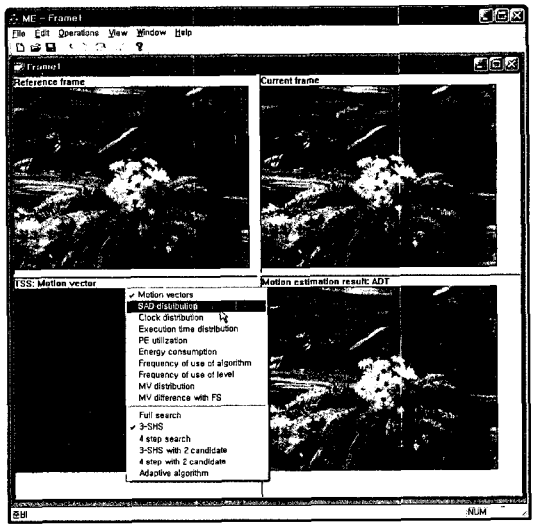


그림 6. 움직임 추정기 시뮬레이터

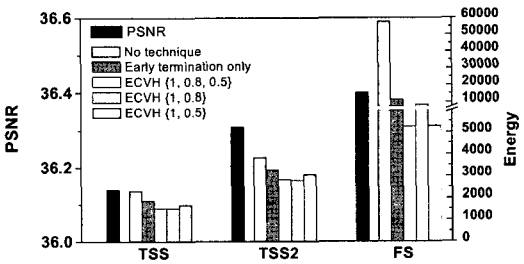
4.1 각 검색 알고리즘의 특성

그림 7은 프레임 시퀀스(frame sequence) 'Miss America'와 'Tempete'의 경우에 각각의 알고리즘의 PSNR과 에너지 소모를 보여주고 있다. 나타난 수치들은 모든 프레임에 대한 값의 평균값이다. 'No technique'은 어떠한 저전력 기술을 적용시키지 않았을 때의 에너지 소모를 나타내며, 'Early termination'은 고정된 동작 주파수와 공급 전압으로 이른 종료를 적용시켰을 때의 에너지 소모를 보여주

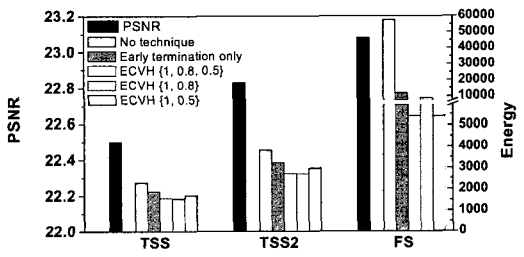
고 있다. 'ECVH'는 주파수레벨이 각각 {1, 0.8, 0.5}, {1, 0.8}, {1, 0.5}인 경우에 대해 단일 알고리즘만을 사용하여 ECVH를 적용시켰을 때의 에너지 소모를 나타내고 있다. 이때 1은 가장 높은 주파수를 나타내며, 0.8과 0.5는 각각 가장 높은 주파수의 80%일 때와 50%일 때를 나타낸다.

아무런 저전력 기법을 사용하지 않았을 때, TSS의 경우는 FS의 4% 정도의 에너지를 소모하며, MC-TSS2의 경우에는 7%의 에너지를 소모하는 것을 알 수 있다. 이른 종료(early termination)를 사용하면 사용하지 않았을 경우에 비해, TSS는 81.3%, MC-TSS2는 85.6%, FS는 20.0% 정도의 에너지를 소모하는 것을 알 수 있다. 주파수 레벨 {1, 0.8, 0.5}와 단일 알고리즘으로 ECVH를 사용할 때 에너지 소모는 사용하지 않았을 때에 비해, TSS는 65.8%, MC-TSS2는 73.3%, FS는 9.1%의 에너지를 소모하는 것을 알 수 있었다. 주파수 레벨 {1, 0.8}의 경우에는 TSS 65.5%, MC-TSS2 72.6%, FS 15.2%의 에너지를 소모하며, 주파수 레벨 {1, 0.5}의 경우에는 TSS 72.6%, MC-TSS2 79.4%, FS 9.2%의 에너지를 소모하는 것을 알 수 있다.

시뮬레이션 결과를 보면 에너지 소모량은 주파수 레벨이 3단계일 때가 2단계보다 전반적으로 낮은 것을 보여주고 있다. 주파수 레벨은 분포가 집중되어 있는 부하(workload)의 비율대로 레벨을 나누는 것이 좋다[4]. 주파수 레벨을 2단계로 나눴을 경우만 고려해 보면, TSS와 2개의 candidate를 갖는 TSS의 multi-candidate 변형(MC-TSS2)의 경우에는 부하의 분포는 최대 부하의 80% 정도에 집중되어 있으며, FS의 경우에는 워크로드 분포가 대부분 최대 부하의 절반 이하에 몰려있다. 따라서 TSS와 MC-TSS2의 경우에 주파수 레벨이 {1, 0.8}일 때가 {1, 0.5}일 때의 에너지 소모가 더 적은 것을 알 수 있으며, FS의 경우에는 그 반대인 것을 알 수 있다. 따라서 주파수 레벨 {1, 0.8, 0.5}을 사용할 때가 가장 좋은 결과를 얻을 수 있다.



(a) Miss America (QCIF)



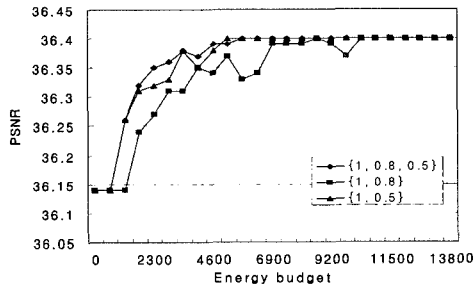
(b) Tempete (CIF)

그림 7. 각 검색 알고리즘 별 시뮬레이션 결과

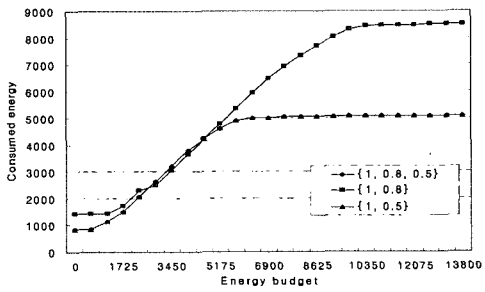
4.2 에너지 할당량에 따른 결과분석

그림 8은 각각의 주파수 레벨의 경우에, 에너지 할당량에 따른 PSNR과 에너지 소모를 보여주고 있다. 에너지 할당량의 증가에 따라, PSNR은 대체적으로 증가하다 포화되는 것을 알 수 있으며, 에너지 소모는 선형적으로 증가하다가 포화되는 것을 알 수 있다. PSNR이 선형적으로 증가하지 않는 것은 고속 탐색 알고리즘들의 상대적인 성능 차이가 항상 일정하지 않기 때문이다.

ECVH 시스템의 경우에 태스크 단위로, 되도록 가장 성능이 좋은 알고리즘을 선택하려는 경향이 있다. 시뮬레이션 결과, 주파수 레벨이 {1, 0.8}인



(a) PSNR



(b) 에너지 소모

그림 8. 주파수 레벨과 에너지 버짓에 따른 ECVH의 특성

경우에 {1, 0.5}인 경우보다 성능이 가장 떨어지는 알고리즘을 많이 선택하지 않지만 가장 좋은 성능을 갖는 알고리즘을 선택하는 비율도 낮은 것을 알 수 있다. 따라서 PSNR의 경우 주파수 레벨이 {1, 0.5}인 경우가 {1, 0.8}인 경우보다 더 좋은 결과를 보여주고 있다. 에너지의 경우 주파수 레벨이 {1, 0.5}인 경우에 낮은 주파수로 동작하는 비율이 {1, 0.8}보다 낮지만, 주파수 레벨이 0.5인 경우가 0.8인 경우보다 훨씬 적은 에너지를 소모하기 때문에 전체적으로 보면 주파수 레벨이{1, 0.5}인 경우가 {1, 0.8}인 경우보다 더 적은 에너지를 소모하는 것을 볼 수 있다. 3단계 주파수를 사용하게 되면 {1, 0.8}인 경우와 {1, 0.5}인 경우의 결과를 서로 보완하므로 PSNR이나 에너지 소모 면에서 가장 좋은 결과를 보여주고 있다.

V. 결 론

본 논문에서는 여러 가지 알고리즘을 적용시킬 수 있고, ECVH를 적용시킬 수 있는 저전력 움직임 추정기 구조를 제안하였다. 이 구조는 에너지 소모를 줄이기 위해 픽셀 재사용, 이른 종료 등의 방법을 사용하며, ECVH를 구현한 전원관리부를 통해 에너지 소모와 검색 성능을 제어하고 있다. 각종 저전력 기술과 주파수 레벨 {1, 0.8, 0.5}와 단일 알고리즘으로 ECVH를 사용할 때, TSS, TSS2, FS의 경우에 각각 66%, 73%, 15%의 에너지만을 소모한다. 또한 TSS, MC-TSS2, FS의 모든 알고리즘으로 ECVH를 사용하면 응용에 따라 에너지 할당량을 달리하여 검색 성능 및 소모하는 에너지를 쉽게 제어할 수 있다.

참 고 문 헌

[1] Vasily G. Moshnyaga, "A New Architecture for Computationally Adaptive Full-Search Block-Matching Motion Estimation," Proc. *IEEE ISCS*, vol.4, pp.219-22, 1999.

[2] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "compensated interframe coding for video conferencing," *Telecommunications Conference*, vol. pp. G5.3.1-G5.3.5, May 1981.

[3] Lai-Man Po and Wing-Chung Ma, "A Novel Four-Step Search Algorithm for Fast Block Motion Estimation," *IEEE TCSVT*, vol.6, no.3, pp.313-17, June 1996

[4] S. Lee, S. Lee, and T. Sakurai, "Journal of Circuits, Systems, and Computers," Vol. 11, No.6, 601-620, 2002

[5] 이성수, "저전력 멀티미디어 통신을 위한 전력 의식 움직임 추정 기법," *한국통신학회논문지* 제29권 제1C호, pp.149-156, 2004. 1.

[6] H. Jong, L. Chen, and T. Chiueh, "Parallel Architectures for 3-Step Hierarchical Search Block-Matching Algorithm," *IEEE TCSVT*, vol.4, no.4, August 1994

[7] R. Richmond, D. Ha, "A Low-Power Motion Estimation Block for Low Bit-Rate Wireless Video," *ISLPED*, pp. 60-63, August 2001

[8] H. Jong, L. Chen, and T. Chiueh, "Accuracy Improvement and Cost Reduction of 3-Step Search Block Matching Algorithm for Video Coding," *IEEE TCSVT*, vol.4, no.1, February 1994

[9] Kuhn P, "Algorithms, Complexity Analysis and VLSI Architecture for MPEG-4 Motion Estimation," *Kluwer Academic*, vol.4, no.1, February 1994

김 응 섭 (Eung-Sup Kim) 학생회원

2004년 2월 숭실대학교 정보통신 전자공학부 졸업

2004년 3월~현재 숭실대학교 전자공학과(석사)

<관심분야> SoC on-chip-bus, MPEG codec 구현

이 찬 호 (Chanho Lee) 정회원

1987년 2월 서울대학교 전자공학과 졸업

1989년 2월 서울대학교 전자공학과(석사)

1994년 6월 UCLA, 전자공학과 (박사)

1994년 8월~1995년 2월 삼성전자 반도체연구소 선임연구원

1995년 3월~현재 숭실대학교 정보통신전자공학부 부교수

<관심분야> 채널코덱의 구현, SoC on-chip-bus, SoC 설계방법론, 3D 그래픽 프로세서 설계, MPEG codec 구현