MPEG-4 AVC를 위한 고속 다해상도 움직임 추정기의 하드웨어 구현

준회원 임 영 훈*, 정회원 정 용 진**

Hardware Implementation of Fast Multi-resolution Motion Estimator for MPEG-4 AVC

Young-hun Lim* *Associate Member*, Yong-jin Jeong** *Regular Member*

요 약

본 논문에서는 다해상도 움직임 추정 알고리즘을 이용하여 모션 벡터를 검색하는 고속 다해상도 움직임 추정기에 대한 하드웨어 구조를 제안한다. 동영상 압축기술인 MPEG-4 AVC 전체 구성 중에서 핵심 부분인 움직임 추정 모듈을 하드웨어로 설계하기 위하여 기본적인 구조를 구성하고 높은 화질로 실시간 부호화를 할 수 있도록 고속 움직임 검색을 위해 특수하게 설계된 램 구조, 메모리 공유, 4화소x4화소 Motion Vector 추출 등과 같은 기술들을 사용하여 전체 움직임 검색기를 구현하였다. 구현된 전체 모듈은 Altera(사)의 Excalibur 디바이스를 이용한 FPGA 구성을 통해 검증하고 최종적으로 Samsung STD130 0.18um CMOS Cell Library를 이용하여 합성 및 검증을 하였다. 이렇게 검증된 구조의 성능은 ASIC으로 구현할 경우 최대 동작 주파수가 약 140MHz이며 QCIF(176화소x144화소) 사이즈 기준으로 초당 약 1100프레임, 4CIF(704화소x576화소) 사이즈 기준으로 초당 약 70프레임의 움직임을 검색할 수 있다. 본 성능은 하드웨어 기반의 MPEG-4 AVC 실시간 부호화기를 설계하기에 적합한 구조임을 보여준다.

Key Words : H.264, MPEG4 AVC, Inter Prediction, Motion Estimation, Motion Compensation

ABSTRACT

In this paper, we propose an advanced hardware architecture for fast multi-resolution motion estimation of the video coding standard MPEG-1,2 and MPEG-4 AVC. We describe the algorithm and derive hardware architecture emphasizing the importance of area for low cost and fast operation by using the shared memory, the special ram architecture, the motion vector for 4 pixel x 4 pixel, the spiral search and so on. The proposed architecture has been verified by ARM-interfaced emulation board using Excalibur Altera FPGA and also by ASIC synthesis using Samsung 0.18 um CMOS cell library. The ASIC synthesis result shows that the proposed hardware can operate at 140 MHz, processing more than 1,100 QCIF video frames or 70 4CIF video frames per second. The hardware is going to be used as a core module when implementing a complete MPEG-4 AVC video encoder ASIC for real-time multimedia application.

* 광운대학교 전자통신공학과 실시간구조 연구실(limyh14@explore.kw.ac.kr), ** 광운대학교 부교수(yjjeong@daisy.kw.ac.kr)
논문번호 : KICS2004-06-041, 접수일자 : 2004년 6월 17일
※본 연구는 IDEC/SIPAC 및 한국소프트웨어진흥원 IT-SoC사업단의 지원으로 이루어졌습니다.

Ⅰ. 서 론

그 동안 다양한 동영상 부호화기를 설계 시 핵심 알고리즘 중에 하나인 움직임 추정 알고리즘들은 여러 가지 방안으로 연구되어왔다. 이 중에서도 다해상도 움직임 추정 방식은 다른 움직임 추정 알고리즘들과 비교하여 우수한 예측 성능과 빠른 계산 속도를 가진다[1]. 다해상도 움직임 추정 알고리즘은 기본적으로 다른 해상도를 포함하는 계층적 모션 벡터 영역으로 구성되고 낮은 해상도 레벨에서 검색된 모션 벡터 후보들은 해상도 변화된 상위 레벨에서 움직임 추정이 되어진 후 마지막으로 해상도 변화된 레벨에 포함된 모션 벡터의 정보를 기반으로 하여 원래 해상도 레벨에서 움직임 추정이 이루어진다. 제안한 움직임 추정 알고리즘은 다해상도 방식과 시공간 관계 특성들을 결합하여 좀더 성능이 향상된 다해상도 움직임 추정 알고리즘을 기반으로 하고 있다. 이러한 방식은 다른 움직임 추정 알고리즘들과 비교하여 볼 때 우수한 성능을 나타내고 있으며 MPEG-1, 2와 같은 동영상 부호화기 뿐만 아니라 MPEG-4 AVC에서도 사용할 수 있도록 제안된 추가적인 기능들을 접목하여 우수한 성능을 가지면서도 실시간으로 동영상을 부호화할 수 있도록 구조를 설계하였다[2].

이후 본 논문의 구성은 다음과 같다. 먼저 2장에서는 다해상도 움직임 추정의 기본 기능에 대해서 설명한다. 3장에서는 FMRME(고속 다해상도 움직임 추정기 : Fast Multi-resolution Motion Estimator)를 위해 추가된 기능에 대해 설명하며 4장에서는 구현된 하드웨어의 전체 구조를 설명한다. 5장에서는 구현된 FMRME 전체 모듈의 동작에 대한 검증 및 성능을 분석하며 마지막으로 6장에서 결론을 맺는다.

Ⅱ. 다해상도 움직임 추정의 기본 기능

일반적으로 모션 벡터의 정밀성은 움직임 추정 성능에 주요한 영향을 미치게 된다. 본 논문에서는 해상도 변화된 레벨에서 단일 모션 벡터 후보가 아닌 다중 모션 벡터 후보들과 모션 벡터 영역에서 공간적인 관계에 대한 모션 벡터 후보를 사용하고, 또한 특수하게 구조된 램을 사용하여 고속 다해상도 움직임 추정기의 구조를 제안하고 이를 이용하여 모션 벡터의 정밀성과 추정을 위한 계산 속도를 높이고자 한다.

다해상도 움직임 추정의 기본 기능은 세 개의 해상도 변화된 레벨로 구성되는데 낮은 해상도 레벨에서는 최소 SAD (Sum of Absolute Difference)값을 기준으로 중간 해상도 레벨에서 계산 되어질 두 개의 모션 벡터 후보들을 얻는다. 중간 해상도 레벨에서는 낮은 해상도 레벨에서 선택된 두개의 모션 벡터 후보와 이전 영상프레임의 원래 해상도 레벨에서 공간적 모션 벡터 4개 값의 평균값으로 얻어진 다른 하나의 후보를 계산하여 원래 레벨의 마지막 최종 벡터의 후보로서 사용한다. 그리고 마지막으로 모션 벡터 후보를 이용하여 원래 해상도 레벨에서 최종 모션 벡터를 선택과 함께 최종적인 SAD 값도 계산하게 된다. 그림 1에서는 최종적인 모션 벡터와 SAD값을 계산하기위한 다해상도 움직임 추정을 위한 순서를 보여준다[1].

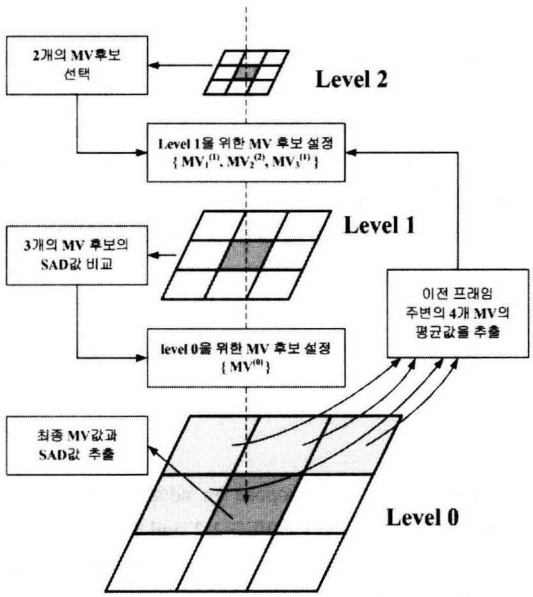


그림 1. 다해상도 움직임 추정 흐름도

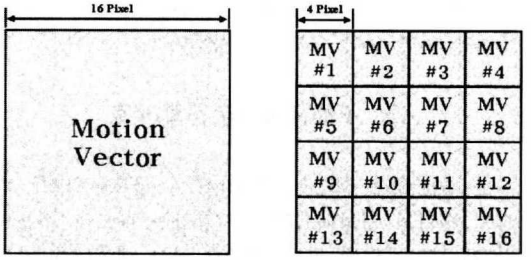
이러한 구조로 이루어진 다해상도 움직임 추정 구조는 모션 벡터의 정밀성을 위해 여러 모션 벡터 후보를 사용하기 때문에 고속의 움직임 추정과 하드웨어에 적합한 구조를 만족시키기 위해서는 다음 장에서 소개되는 추가적인 기능들이 필요하게 된다. 추가적인 기능 등을 이용하여 구현된 고속 다해상도 움직임 추정기는 MPEG-1, 2에서의 16화소x16화소의 모션 벡터의 움직임 추정뿐만 아니라 MPEG-4 AVC와 같은 동영상 부호화기를 위한 진보된 움직임 추정 모드(4화소x4화소 ~ 16화소x16화소)까지 제공한다.

III. FMRME를 위해 추가된 기능

본 논문에서 제안한 고속 다해상도 움직임 추정기는 다해상도 움직임 추정기의 기본 기능에서 MPEG-4 AVC 및 고속 동영상 부호화기에서도 사용할 수 있도록 제안된 기능들을 추가하여 실시간 부호화가 가능하도록 알고리즘의 구조를 개선하였다. 추가된 기능으로는 MPEG-4 AVC에서 사용할 수 있도록 4화소x4화소 단위의 모션 벡터 추출기능과 가변 폭 데이터 버스 방식, 메모리 재사용을 이용한 데이터 입력 방식, 나선형 방식의 검색, 가상 주소(Virtual Address) 방식, 고속 다해상도 움직임 추정기를 위한 특별한 램 구조의 사용이 있다. 이러한 기능들이 추가됨으로서 실시간으로 부호화가 가능하게 할 수 있는 고속의 움직임 추정을 할 수 있게 된다.

3.1 4화소x4화소 단위의 모션 벡터 추출기능

일반 MPEG-1, 2등과 같은 동영상 부호화기 뿐만 아니라 MPEG-4 AVC에서도 사용할 수 있도록 4화소x4화소에서부터 16화소x16화소까지의 다양한 모션 벡터를 추정할 수 있는 구조로 설계되어 있으며 레지스터 설정에 의해 사용하고자 하는 모션 벡터를 추정하기 위한 범위의 크기를 설정할 수 있다. 아래 그림 2에서는 대표적으로 사용되어지는 매크로블록 단위의 모션 벡터 추정 범위를 나타내었다.



(a)16화소x16화소 모드 (b)4화소x4화소 모드

그림 2. 다양한 크기의 모션 벡터 추정범위

3.2 FMRME를 위한 특별한 램 구조의 사용

(1) 해상도 변환기(Resolution Converter)

원 해상도 레벨의 영상데이터를 입력받아 메모리에 저장할 시에 해상도 변환기를 사용하여 추가의 클록 사용 없이 레벨 1과 레벨 2의 해상도 변화된 데이터를 동시에 각각의 레벨에 맞은 메모리에 저장하는 방식이다.

(2) 입력 주소 생성기(Input Address Generator)

해상도 변환기에 의해 입력되는 영상 데이터를 출력 주소 생성기에서 원활하게 사용될 수 있도록 그림 3과 같은 순서로 각각의 16개의 메모리 블록에 저장하는 기능이다.

(3) 출력 주소 생성기(Output Address Generator)

움직임 검색을 위해 데이터를 각 레벨의 비교기로 데이터를 출력할 시에 BSU(Basic Search Unit)에서 단일 클록에 16개의 메모리 블록에서 영상데이터를 입력받을 수 있는 기능을 가지고 있다. 즉, 동시에 16개의 8비트 영상 데이터를 추출할 수 있는 기능을 가지고 있으며 움직임 추정시간을 줄이기 위한 중요한 기능 중의 하나이다. 그림 3에서는 입력되는 영상데이터의 순서를 나타내며 좌측 상단의 16개 영상 데이터 단위로 동시에 출력이 가능하다.

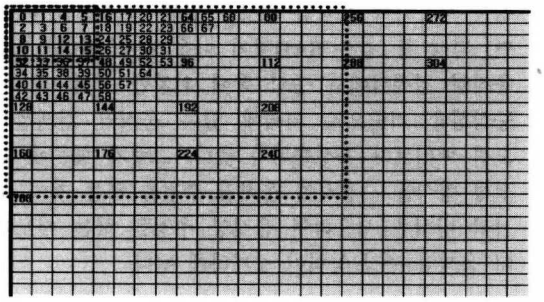


그림 3. 입력된 영상데이터의 순서

3.3 메모리 재사용을 이용한 데이터 입력

메모리 재사용 방식이란 입력받은 영상 데이터 중에서 매크로블록 단위로 움직임 추정 후 다음에 입력받을 영상 데이터가 동일할 시 새로 영상 데이터를 입력받는 것이 아니라 메모리의 위치를 이동하여 영상 데이터의 입력 횟수를 줄이는 방식이다. 메모리 재사용 방식을 사용하지 않을 시에 각 매크로블록마다의 움직임 추정 시 9개의 매크로블록을 읽어 와야 하며 아래 그림 4와 같이 메모리 자체에서 데이터의 이동이 필요하다. 그러나 메모리안의 데이터가 이동하기 어려우므로 그림 5와 같이 메모리 재사용 방식으로 사용하여 내부적으로 매크로블록 포인터를 이용하여 영상데이터가 실제 메모리 이동을 하지 않고도 움직임 추정을 할 수 있도록 하였다.

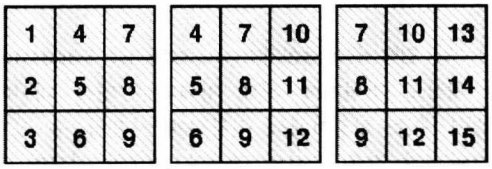


그림 4. 일반 영상 데이터 사용 방식

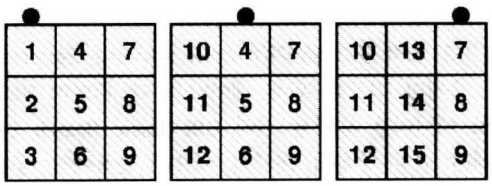
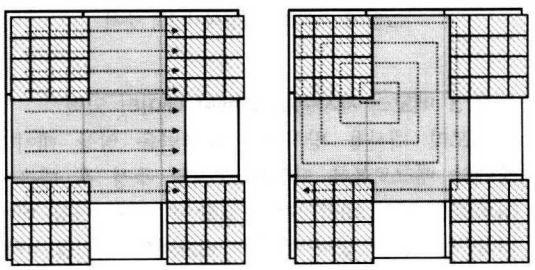


그림 5. 매크로블록 포인터를 사용한 방식

3.4 나선형 방식의 검색

각 해상도 레벨에서의 움직임 추정 시에 sSAD값을 비교하기 위한 일반적인 검색 순서는 그림 6의 (a)와 같은 주사선(Raster) 방식을 사용한다. 그러나 검색 순서를 그림 6의 (b)와 같이 나선형(Spiral) 방식을 사용하여 움직임 추정을 위한 검색을 하게 되면, 검색 중 같은 SAD값이 나올 경우 일반적으로 움직임 검색점에서 중심 쪽에 위치한 모션 벡터가 중심점을 기준으로 외곽에 있는 움직임 검색점의 모션 벡터보다 우선순위가 높다. 그렇기 때문에 나선형 방식을 사용하여 움직임 추정을 하는 것이 주사선 방식을 사용하여 움직임 추정을 하는 것보다 더 움직임 추정 능력이 증가하게 된다.



(a) 주사선 방식 (b) 나선형 방식

그림 6. 움직임 추정 시의 검색 순서

3.5 가상 주소 방식

해상도 변화된 레벨 2에서 추정되어진 모션 벡터 후보들을 사용하여 레벨 1과 레벨 0에서 움직임 추정을 할 때에 영상 데이터의 주변 8개의 매크로블록의 끝부분에서의 검색 오류를 피하고 검색력을 높이기 위해 그림 7과 같이 가상의 주소방식을 사용한다. 이 방식을 사용하면 실제 메모리의 증가 없이 각 레벨간의 움직임 추정위치에 따른 검색능력이 증가하게 된다.

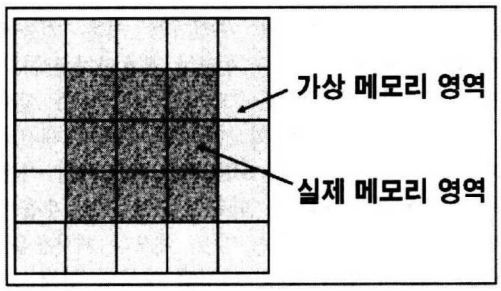


그림 7. 검색 메모리의 가상 주소 방식

3.6 가변 폭 데이터 버스 사용

기본적인 영상데이터 하나의 화소를 기준으로 영상 데이터를 입력할 시에는 일반적으로 8비트 데이터 버스를 사용하게 된다. 제안된 구조에서는 이러한 일반적인 움직임 추정기에서도 기본적으로 사용할 수 있도록 하였으며 추가적으로 32비트 데이터 버스를 사용할 수 있도록 기능을 추가하였다. 32비트 데이터 버스 방식이란 8비트의 영상 데이터 4개를 한번에 묶어서 전송하는 방식으로써 32비트 프로세서를 사용하는 동영상 압축 시스템에서 움직임 추정 Accelerator 기능으로 사용 시에 8비트로 사용하는 것보다 유리하다. 이러한 32비트 데이터 버스 방식을 사용하면 8비트 데이터 버스 방식보다 영상 데이터의 입력 속도가 4배 증가하게 되어 움직임 추정을 위한 동작 속도가 향상된다.

Ⅳ. FMRME의 전체 구조

본 논문에서 제안한 FMRME의 구조는 크게 해상도 변화된 영상데이터를 저장하는 메모리 부분과 각 해상도 레벨에 따라 저장된 영상 데이터를 비교하는 비교기 부분으로 나누어져 있다. 세부적으로 살펴보면 영상 데이터 및 모션 벡터, SAD값 등과 같은 데이터를 입출력하는 PCI 또는 ARM 인터페이스부분과 매크로블록 단위의 영상 데이터를 저장하는 특별한 구조의 메모리 입출력 부분, 16 PE(Process Element)로 이루어져 단일 클록에 16개 영상 데이터의 SAD를 계산할 수 있는 BSU, 레벨 0, 1, 2의 비교기 모듈, 그리고 전체 흐름을 제어하는 전체 제어기(Fast Multi Resolution Motion

Search Controller)로 이루어져 있다.

움직임 측정을 위해 최초로 수행되어져야 할 순서는 PCI 또는 ARM 인터페이스를 통하여 영상데이터와 이전 모션 벡터의 평균값을 입력받는 것이다. 이때 영상데이터를 입력받을 시 해상도 변환기를 통해 추가적인 시간소요 없이 각 해상도 변화된 값이 각각의 메모리에 저장된다. 이렇게 저장된 데이터는 전체 제어기에 따라 각 레벨의 계산 순서에 의해 BSU에 입력되어진다. BSU에 입력될 때 출력 주소 생성기에 의하여 16개의 데이터가 동시에 추출되어 한번에 계산이 되어진다. BSU에서 계산된 SAD값은 각 레벨의 비교기 모듈로 입력되어지며 비교기를 통하여 나온 데이터에 의해 다음 계산이 진행되어진다. 이러한 방식으로 계속 순환하여 계산하게 되면 최종적으로 마지막 SAD값과 모션 벡터값이 추출된다. 이와 같은 구조를 가지고 있는 FMRME의 전체 구조는 그림 8과 같다.

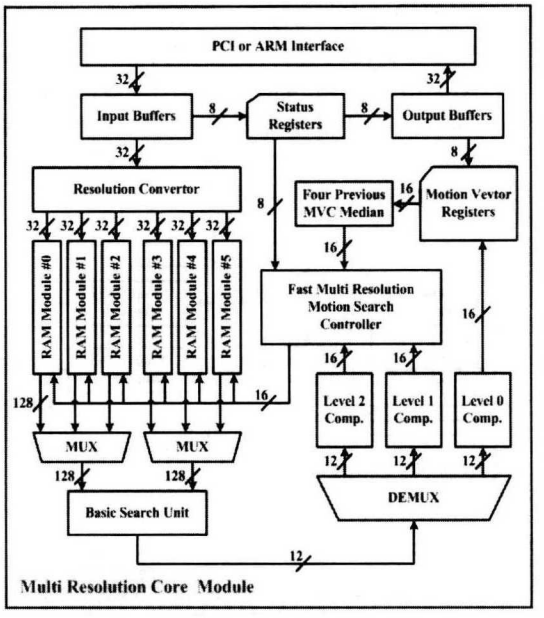


그림 8. 다해상도 움직임 추정기 전체구조

4.1. FMRME를 위한 메모리 구조

FMRME를 구현하기 위해서 가장 중요시 되어지는 부분이 메모리의 구조이다. 움직임 검색을 위해서는 움직임 검색을 통하여 모션벡터와 SAD값을 추출하기 위한 검색 시간도 중요하지만 전체적인 움직임 추정 시간에서 실질적인 지연시간을 영상데이터의 입출력이 차지하고 있기 때문에 영상 데이터의 읽고 쓰는 속도가 무엇보다도 중요하다. 본 논문에서는 이러한 문제를 해결하기 위하여 크게 두 가지의 방법을 사용하였다.

첫번째는 하나의 매크로블록 단위의 영상데이터를 메모리에 저장 시에 추가의 클록 사용 없이 해상도 변화된 값을 저장하는 기법을 사용하였다. 이때 사용한 방법으로는 입력 주소 생성기와 간단한 계산 모듈과 레지스터를 사용하여 입력되는 데이터를 특별한 순서에 의해 각각의 메모리 블록에 저장되도록 하였다.

두번째로는 움직임 검색을 위해 데이터를 읽어올 때 한번에 BSU 단위로 읽어 올 수 있도록 동시에 16개의 8비트 데이터를 추출할 수 있는 기법을 사용하였다. 16개의 영상 데이터를 동시에 추출하기 위해서는 각각 메모리 블록에 저장된 영상 데이터를 전체 제어기에 의해 한 클록에 꺼내어 계산하도록 하였다. 그림 9는 FMRME에서 사용되는 특별한 메모리 구조를 나타낸다.

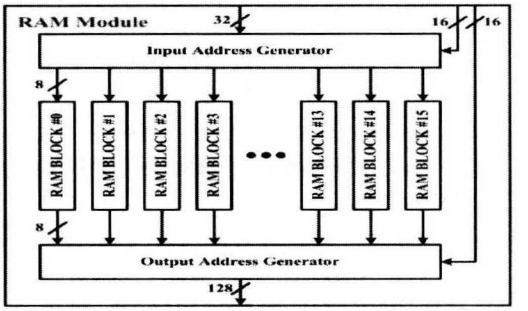


그림 9. FMRME의 메모리 구조

4.2 레벨 2에서의 검색 구조

기본적으로 메모리 모듈에서부터 출력되는 16개의 8비트 영상 데이터는 BSU를 통하여 SAD값이 계산되어진다. 이렇게 계산된 값을 이용하여 움직임 추정에 사용하는 검색 구조로 이루어져 있다. 검색영역은 총 144화소(=12화소x12화소)로 이루어져 있고, 검색점은 81포인트(9포인트x9포인트)로 이루어져 있다. 최소의 SAD값을 찾기 위하여 전역 검색 방식으로 총 81개의 위치를 검색한다. 하나의 검색점은 BSU에 의하여 계산되며 두개의 최소 SAD값를 가진 위치를 검색하여 중간 해상도인 레벨 1에서 사용하기 위해 레지스터에 저장한다. 그림 10은 레벨 1에서 사용될 모션 벡터 후보를 검색하기 위한 비교기 구조이다.

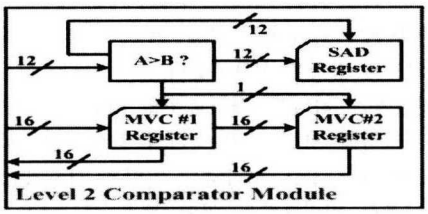


그림 10. 레벨 2의 검색기 구조

4.3 레벨 1에서의 검색 구조

레벨 2의 구조와 마찬가지로 기본적으로 BSU에서 나온 값을 이용하여 계산하는 구조로 이루어져 있다. 레벨 2에서 검색된 2개의 후보의 위치를 레벨 1의 좌표에 해상도 변화 후 같은 위치 값을 찾아 새로운 검색 영역을 지정하며, 또 하나의 후보는 이전 프레임의 주변 모션 벡터 값의 평균값을 취하여 3번째 후보로서 입력받게 된다. 모두 3개의 모션 벡터 후보를 검색하게 되며, 검색영역은 계산에 사용되는 BSU의 구조가 16개의 PE(Process Element)로 구성되어 있기 때문에 8화소x8화소의 크기를 가지고 있는 각각의 모션 벡터 후보들을 4부분으로 나누어서 검색하게 된다. 즉, 총 검색영역은 588화소((7화소x7화소)x4영역x3후보)로 이루어져 있고, 검색점은 192포인트((4포인트x4포인트)x4영역x3후보)로 이루어져 있다.

레벨 1에서의 최종 후보를 찾기 위해서는 4개의 위치에 대한 SAD값을 모두 합하여 저장한 후 3개의 모션 벡터 후보 값들을 비교하게 되며 최소 SAD값으로 검색되어진 모션 벡터 최종후보의 위치를 Level 0에서 사용하기 위해 레지스터에 저장한다. 그림 11은 레벨 0에서의 최종적으로 사용될 모션 벡터 후보를 검색하기 위한 비교기 구조이다.

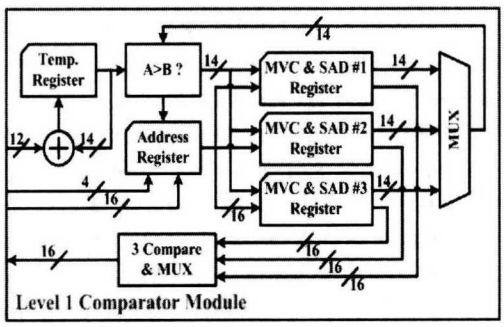


그림 11. 레벨 1의 검색기 구조

4.4 레벨 0에서의 검색구조

레벨 0에서의 검색 구조도 레벨 1과 마찬가지로 기본적으로 16개의 PE로 구성되어 있는 BSU를 이용하여 계산된 SAD값을 비교하여 최종적인 모션 벡터와 SAD값을 구하게 된다. 레벨 1에서 검색된 최종 모션 벡터 후보의 위치를 레벨 0의 좌표에 해상도 변화 후 같은 위치 값을 찾아 새로운 검색 영역으로 지정하게 된다. 이때 검색영역은 16화소x16화소의 크기를 가지고 있으므로 16개의 부분으로 나누어 계산하게 된다. 그러므로 총 검색영역은 784화소((7화소x7화소)x16영역)로 이루어져 있고 검색점의 수는 256포인트((4포인트x4포인트)x16영역)가 된다.

이러한 검색영역과 검색점을 전체 움직임 추정 제어기에 의해 검색을 한 후 최종적인 모션 벡터에 대한 위치 값을 추출하여 레지스터에 저장한다. 이때 최종 위치에 대한 16개의 위치 값을 더하여 레지스터에 저장하는 방법과 따로 16개의 모션 벡터의 위치 값을 레지스터에 저장하는 방법에 의해 16화소x16화소 모드와 4화소x4화소 모드로 나누어진다. 그림 12는 레벨 0의 검색기 구조 중에 16화소x16화소 모드를 나타내며 그림 13은 4화소x4화소 모드의 검색기 구조를 나타낸다.

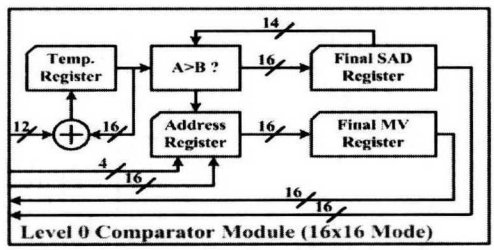


그림 12. 레벨 0의 검색기 구조 (16x16 모드)

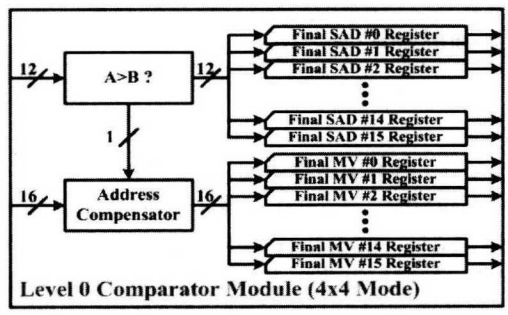


그림 13. 레벨 0의 검색기 구조 (4x4 모드)

4.5 BSU(Basic Search Unit)의 구조

고속 움직임 추정기에서 사용되는 BSU는 16개의 PE로 구성되어 있으며 현재 프레임(Current Frame)의 화소값과 이전 프레임(Reference Frame)의 화소값의 차이값을 절대값으로 변환하여 16개의 값을 모두 더하는 방식으로 계산되어진다. 일반적으로 CPA(Carry Propagation Adder)를 사용하여 16개의 PE를 더하게 되면 최장 지연 시간(Critical Path)이 길어진다. 그러나 지연 시간을 최소화하기 위해 일반 CPA를 사용하지 않고 CSA(Carry Save Adder)와 CLA(Carry Lookahead Adder)를 사용하여 최장 지연시간을 줄일 수 있도록 구성하였다. 그림 14는 BSU의 구조를 나타내고 있다.

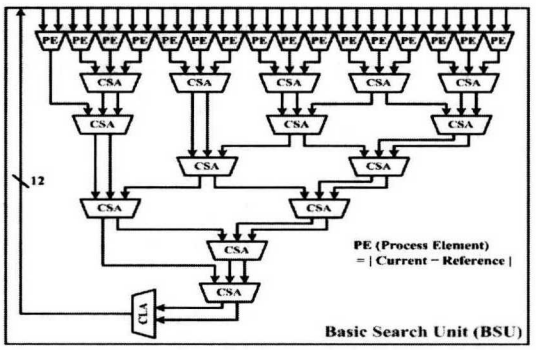


그림 14. BSU의 내부 구조

V. 동작 검증 및 성능 분석

5.1 FMRME의 동작 검증

제안된 구조를 사용하여 구현된 FMRME에서 동작을 검증하기 위해 JVT(Joint Video Team)에서 제공한 JM6.0을 사용하여 몇 가지 움직임 추정 알고리즘에 대한 PSNR(Peek-to-peek Signal Noise Ratio)값을 비교하였고 구현된 모듈을 사용하여 움직임 추정을 하기 위한 동작 속도에 대하여 검증을 하였다.

먼저 JM6.0을 이용하여 PSNR값과 하나의 영상 프레임에서 인트라 매크로블록들의 검출되는 빈도를 비교 검증하였다[6]. 이중 PSNR값은 부호화시에 원 영상과의 화질차를 비교하는 척도이며 영상 프레임 내의 인트라 매크로블록들의 검출 빈도는 움직임 추정을 제대로 수행했는지에 대한 비교 척도이다. 테스트를 위해 JM6.0의 기존 소스에 기본적으로 사용하는 고속 전역 검색 방식과 추가적으로 삽입한 기본 전역 검색 방식, 다해상도 검색 방식을 이용하여 Akiyo, Foreman, Stefan의 3가지 동영상에 대한 테스트를 수행하였으며 결과 데이터를 토대로 표 1에 비교한 결과를 나타내었다. 표에서 알 수 있듯이 움직임 추정 알고리즘 중에 PSNR값이 가장 높은 전역 검색과 비교하여 차이가 거의 없음을 나타내고 있다.

FMRME에서의 동작속도를 분석하여 보면, 동작속도를 계산하기 위해서는 크게 메모리 클록과 시스템 클록으로 나누어 계산하게 되는데 메모리 클록은 영상 데이터를 프레임 메모리로부터 입력받기 위한 클록이고, 시스템 클록은 입력된 영상 데이터의 값을 이용해 움직임 추정에 쓰이는 클록이 된다. 메모리 클록수는 이전 영상 데이터, 현재 영상 데이터, 이전 모션 벡터를 입력 받기위한 클록의 수가 되고, 시스템 클록수는 움직임을 추정하기 위한 레벨 2, 레벨 1, 레벨 0의 움직임 추정의 검색 클록수가 된다.

표 1. JM6.0을 이용한 테스트 결과

	PSNR[dB]			인트라 매크로블록		
기준 영상	Akiyo	F.man	Stefan	Akiyo	F.man	Stefan
기본 전역 검색	37.15	34.69	31.86	0	2.1	7.5
고속 전역 검색	37.15	34.78	31.85	0	2.2	8.1
다해상도 검색	37.15	34.75	31.85	0	2.1	8.5
QCIF 영상 기준, 10개 프레임 평균값						

움직임 추정에 사용되는 클록의 수를 살펴보면 기본적으로 32비트 데이터 버스 방식을 사용한다고 가정할 때 메모리 재사용 방식을 사용하지 않고 영상 데이터를 입력 받기 위한 메모리 클록은 642클록이 소요된다. 그러나 메모리 재사용 방식을 사용할 시에는 하나의 영상프레임에서 각 매크로블록 행중에서 첫 번째 매크로블록만 642클록이 소요되고 이후 매크로블록의 메모리 클록은 258클록만이 소요된다. 움직임 추정을 위한 시스템 클록은 메모리 재사용 방식과 관계없이 594클록이 소요된다. 이때 16화소x16화소의 모션 벡터와 4화소x4화소의 모션 벡터를 추출하는 시간은 동일하다. 그림 15에서는 움직임 추정을 위한 메모리 클록과 시스템 클록의 수를 나타내었다.

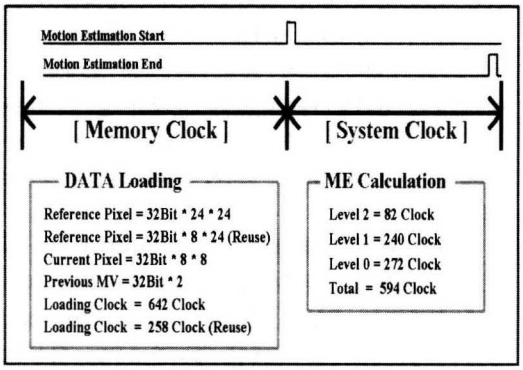


그림 15. 움직임 추정을 위한 클록의 수

5.2 FMRME에서의 성능 분석

구현된 모듈들의 설계 검증을 위해 FPGA와 ASIC의 두 가지 방법을 사용하여 결과 값을 분석하였다. 첫번째로는 Altera사의 ARM922T를 내장한 디바이스인 Excalibur계열의 EPXA10F1020C2를 이용하여 인터페이스를 구성하고 구현한 모듈을 리눅스 환경에서 테스트 할 수 있는 프로그램을 작성하여 구현된 고속 움직임 추정기의 동작을 검증하였다. 이렇게 FPGA로 구현 시 결과 값에 따른 최대 동작 주파수는 14MHz이며 대략 19%의 로직 리소스와 8%의 메모리 비트가 사용되어진다.

두번째로는 Samsung STD130 0.18um CMOS Cell Library와 Synopsys사의 Design Analyzer를 이용하여 합성 결과 값을 분석하였다. 구현된 전체 모듈을 ASIC Library를 이용하여 합성한 결과, 측정된 최장 지연 경로(Critical Path)는 7.01ns이며 최대 동작 주파수는 약 140MHz가 된다. 이러한 동작 속도에 따라서 구현된 하드웨어 사이즈는 로직 리소스는 약 24K게이트가 사용되며 메모리는 3.3KByte가 사용되었다. 그러나 다른 MPEG-4 AVC 모듈과의 유연한 동작을 위해 100MHz로 최적화시켜 합성하게 되면 하드웨어 사이즈 중에 로직 리소스가 약 21.5K게이트로 줄어들게 된다. 표 2는 전체 모듈에서 각 모듈별로 구현된 게이트 사이즈와 동작 최장 지연시간을 나타내었다.

구현된 모듈들의 성능에 의해서 움직임 추정 시 계산되어지는 속도를 살펴보게 되면, 메모리 재사용 방식을 사용하지 않을 때 하나의 매크로블록의 모션 벡터와 SAD값을 추출하기 위한 속도는 위에서 나타낸 바와 같이 메모리 클록과 시스템 클록을 동일하게 사용하고 동작 클록을 100MHz로 사용할

표 2. 구현된 모듈 성능(삼성 STD130 0.18um공정)

모듈	Area [gates]	
	약 100MHz 동작 시(9.80ns)	약 140MHz 동작 시(7.01ns)
Basic Search Unit	4,290	5,318
Comparators	8,113	8,533
Controller	5,844	7,877
Ram Generator	3,098	2,735
ARM Interface	158	159
Total	21,519	24,058

시 1236 (메모리 클록 + 시스템 클록 = 642 + 594)클록이 소요된다. 초당 30프레임의 QCIF (176화소 x 144화소) 크기의 영상을 기준으로 성능을 분석하여 보면 99개의 매크로블록 기준으로 계산하여 보면 약 0.037초에 계산이 됨을 알 수 있다. CIF(352화소 x 288화소) 크기의 영상을 기준으로 30프레임의 영상을 검색하는데 걸리는 시간이 약 0.147초에 계산이 됨을 알 수 있다.

메모리 재사용 방식을 사용하여 QCIF 크기의 영상을 기준으로 99개의 매크로블록의 모션 벡터와 SAD값을 추출할 때에는 사용되어지는 메모리 클록 수가 줄어들기 때문에 더 빠른 검색이 가능하다. 즉 메모리 재사용 방식을 사용하면 99개의 매크로블록 중에 9개는 642클록의 메모리 클록이 사용되어지고 나머지 90개는 258개의 메모리 클록이 사용되어진다. 즉 전체 매크로블록의 메모리 클록은 28,998클록이 소요되며 메모리 재사용 방식을 사용하지 않을 때의 63,558클록과는 속도 면에서 많은 차이를 보이게 된다. 즉 QCIF 크기의 영상에서 30프레임을 검색하는데 걸리는 시간이 약 0.026초이며 QCIF의 16배의 크기를 가진 4CIF 크기의 영상에서는 약 0.416초가 소요된다. 이러한 속도는 MPEG-4 AVC 부호화기에서 영상 데이터를 실시간으로 부호화할 수 있기에 충분한 성능을 보여준다.

다른 몇 가지의 움직임 추정 알고리즘들과 제안된 구조의 성능을 비교하여 보면 대표적으로 두 가지 방식의 FS(Full Search) 알고리즘을 적용한 모듈들과의 성능 비교에서 제안된 FMRME의 성능이 기본적인 움직임 추정 알고리즘인 FS보다 하드웨어 사이즈나 처리량 면에서 월등한 성능을 나타내고 있다. 또한 MRMCS 알고리즘을 사용하여 구현된 모듈과 비교하여 보면 메모리 재사용을 위해 메모리의 양이 추가되었으나 영상데이터 입력속도 및 하드웨어 사이즈 대비 처리량에서 더 좋은 성능을

보이고 있다. 표 3에서는 하나의 매크로블록을 기준으로 하여 본 논문에서 제안된 FMRME의 구조와 FS, MRMCS와의 성능을 비교하였다.

표 3. 다른 움직임 추정 알고리즘과의 성능 비교

구조	FS [3]	FS [4]	MRMCS [5]	제안된 구조
입력 데이터 폭	24	48	8	32(8)
PE의 갯수	256	64	5	16
입력속도 [클록]	768	384	3360	642 (2562)
처리량 [클록/블록]	1024	4096	2640	594
게이트 수 사이즈	192.2K	N/A	25K	21.5K
메모리	-	-	160Byte	3.3KByte

VI. 결 론

본 논문에서는 MPEG-4 AVC를 위한 고속 다해상도 움직임 추정 알고리즘의 구조를 기술하고 하드웨어 설계를 위한 구조를 제안하고 구현하였다. 기본적인 다해상도 움직임 추정 알고리즘을 위한 기본 구조를 구성하고 높은 화질로 실시간 부호화를 할 수 있도록 고속 움직임 추정을 위해 특수하게 설계된 램 구조, 메모리 재사용 방식, 4화소x4화소 모션 벡터 예측, 가상 주소 방식, 나선형 검색, 가변 버스 폭 방식 등과 같은 추가적인 기술들을 사용하여 성능향상을 꾀하였다. 이렇게 설계된 고속 다해상도 움직임 추정기는 다른 움직임 추정 알고리즘보다 하드웨어 사이즈나 처리량에서 우수한 성능을 나타내고 있다.

제안된 전체 모듈은 다른 움직임 추정 알고리즘과 비교하여 적은 하드웨어 리소스와 동작 클록으로 구현할 수 있고, 또한 실시간으로 동영상 부호화가 가능한 성능을 가지고 있기 때문에 MPEG-4 AVC를 위한 고속 움직임 추정기로 사용하기에 우수한 구조라고 할 것이다. 현재는 전체 MPEG-4 AVC의 부호화기 부분 중에서 움직임 보정, 인트라 예측, 엔트로피 코딩, 루프 필터 등의 각 모듈들을 설계 중에 있으며 향후 전체 구조를 하드웨어로 구현할 예정이다.

참 고 문 헌

[1] B. Song and J. Ra, "A fast multi-resolution block matching algorithm for motion estimation", Signal Processing : Image Communication, vol.15, pp. 799-810, 2000.

[2] J. Chalidabhongse and C. Kuo, "Fast motion vector estimation using multi-resolution spatio temporal correlations", IEEE Trans. on Circuits & Systems Video Technology, vol. 7, no. 3, pp. 477-488, June 1997.

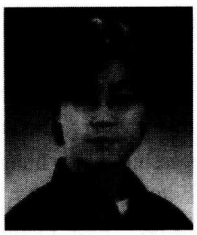
[3] L. De Vos and M. Stegherr, "Parameterizable VLSI architectures for the full-search block-matching algorithm," IEEE Trans. on Circuits & Systems, vol. 36, pp. 1309-1316, Oct. 1989.

[4] K. M. Yang, M. T. Sun, and L. Wu, "A family of VLSI designs for the motion compensation block matching algorithm," IEEE Trans. on Circuits & Systems, vol. 36, pp. 1317-1325, Oct. 1989.

[5] H. Lee, K. Lim, B. Song and J. Ra, "A Fast Multi-Resolution Block Matching Algorithm and its LSI Architecture for Low Bit-Rate Video Coding", IEEE Trans. on Circuits Systems Video Technology, vol. 11, no.12, December 2001.

[6] JVT H.264 Reference Software version 6.0, ftp://ftp.imtc-files.org/jvt-experts/, 2002.

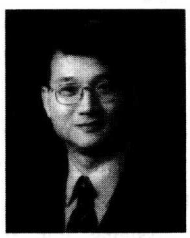
임 영 훈(Young-hun Lim) 준회원



2001년 1월 : 대진대학교 전자공학과 졸업
2001년 1월~2003년 2월 : 동서전자 기업부설연구소 연구원
2003년 2월~현재 : 광운대학교전자통신공학과 석사과정

<관심분야> 영상처리, 제어 시스템 설계, SoC 설계, 임베디드 시스템

정 용 진(Yong-jin Jeong) 정회원



1983 2월 : 서울대학교 제어계측공학과 졸업
1983년 3월~1989년 8월 : 한국전자통신연구원
1991년 5월 : 미국 UMASS 전자전산공학과 석사
1995년 2월 : 미국 UMASS 전자전산공학과 박사
1995년 4월~1999년 2월 : 삼성전자 반도체 수석연구원
1999년 3월~현재 : 광운대학교 전자통신공학 부교수

<관심분야> 컴퓨터 연산 알고리즘, SoC 설계, 무선통신, 정보보호, 임베디드 시스템