일반논문-11-16-2-09

H.264 비디오 코덱을 위한 고속 움직임 예측기의 하드웨어 구조

임 정 훈[a], 서 영 호[a], 최 현 준[b], 김 동 욱[a]‡

A New Hardware Architecture of High-Speed Motion Estimator for H.264 Video CODEC

Jeong-Hun Lim[a], Young-Ho Seo[a], Hyun-Jun Choi[b], and Dong-Wook Kim[a]

요 약

본 논문에서는 H.264/AVC 인코더에서 가장 많은 연산 시간이 소요되는 움직임 추정(motion estimation, ME) 동작을 위한 하드웨어의 구조를 제안하고 IP(intellectual property) 형태로 구현하였다. 고속 움직임 추정기의 구조는 버퍼(buffer), PU 어레이(processing unit array), SAD 선택기(SAD selector), MV 생성기(motion vector generator) 등으로 구성되어 있다. PU 어레이는 16개의 PU로 구성되어 있고, 각각의 PU는 16개의 PE(processing element)로 이루어져 있다. 제안한 하드웨어의 동작적인 특징은 외부메모리 접근량을 줄이기 위해 현재와 참조프레임의 데이터를 재사용한다는 것과 SAD연산을 수행할 때 클록의 손실 없이 계산을 할 수 있다는 것이다. 구현한 고속 움직임 추정기는 Altera 사의 FPGA인 StatixⅢ EP3SE80F1152C2에서 3%의 자원을 사용하였고, 최대 동작주파수는 446.43MHz이었다. 따라서 구현한 하드웨어는 1080p 영상을 최대 50fps로 처리할 수 있다.

Abstract

In this paper, we proposed a new hardware architecture for motion estimation (ME) which is the most time-consuming unit among H.264 algorithms and designed to the type of intellectual property (IP). The proposed ME hardware consists of buffer, processing unit (PU) array, SAD (sum of absolute difference) selector, and motion vector (MVgenerator). PU array is composed of 16 PUs and each PU consists of 16 processing elements (PUs). The main characteristics of the proposed hardware are that current and reference frames are re-used to reduce the number of access to the external memory and that there is no clock loss during SAD operation. The implemented ME hardware occupies 3% hardware resources of StatixⅢ EP3SE80F1152C2 which is a FPGA of Altera Inc. and can operate at up to 446.43MHz. Therefore it can process up to 50 frames of 1080p in a second.

Keyword : Data reuse, H.264/AVC standard, Motion Estimation, memory access reduction, Inter prediction

a) 광운대학교
Kwangwoon University
b) 안양대학교
Anyang University
‡ 교신저자 : 김동욱 (dwkim@kw.ac.kr)
※본 연구는 지식경제부 및 한국산업기술평가관리원의 산업융합원천기술개발사업(정보통신)의 일환으로 수행하였음. [KI002058, 대화형 디지털 홀로그램 통합서비스 시스템의 구현을 위한 신호 처리 요소 기술 및 SoC 개발]
・접수일(2010년10월11일),수정일(2011년2월11일),게재확정일(2011년3월10일)

Ⅰ. 서 론

현대인들은 점점 더 보다 나은 영상을 원하고 있다. 더욱이 최근 들어서 3DTV, UHDTV등의 고화질 영상들을 더욱 빠르게 압축하고 처리하기 위해서는 움직임 추정의 연산을 보다 빠르게 처리할 수 있는 기술 혹은 부품이 필요하다.

H.264의 움직임 추정은 부호화기 전체의 약 95% 이상을 움직임 찾기 연산과 메모리 접근에 사용하고 있다. 움직임 추정에 있어서 SAD 연산량은 영상의 크기와 검색윈도우의 크기와 블록의 크기에 따라 증가한다. 연산량을 줄이기 위하여 기존에 제안된 연구 방법들은 연산시간과 메모리 접근을 줄이기 위하여 고속의 병렬 연산 구조에 관한 연구들과, 병렬 연산 구조의 메모리대역폭을 줄이기 위한 중간버퍼관리, 검색윈도우관리, 움직임추정을 위해 데이터 재사용을 이용한 하드웨어구조, 다중 참조프레임의 정보를 재사용하는 방법 등이 제안되고 있다[1][2][3].

기존의 다중참조프레임을 이용한 움직임 추정 방법들은 참조프레임의 개수만큼의 On-chip 메모리가 필요로 하게 되어 하드웨어로 구현하기에는 비효율적이었다. 특히 외부메모리의 접근은 크게 3가지의 요소에 큰 영향을 받게 되는데, 참조프레임, 탐색범위의 중심점, 그리고 탐색범위의 크기가 있다. 고화질의 영상을 처리함에 있어, 이러한 3가지의 요소들을 하드웨어로 구현하기에는 복잡도가 높아져 구현하기에 어려움이 많다[4].

이러한 문제점을 줄이려는 기존의 연구들이 제안한 하드웨어 구조들의 가장 큰 단점은 데이터 재사용량이 매우 적었다[5][6][7][8][9]. 특히 Full Search Block-Matching 의 구조를 가지고 최소의 메모리 대역폭이 요구되는 구조들이 제안되었으나 움직임 추정에 있어 7가지의 가변블록들을 처리 할 수 없고 고정된 블록 크기만 처리 할 수 있었다[10].

본 논문에서는 위에서 제기된 문제점들을 보완, 발전하기 위하여 참조프레임의 데이터 처리에서 발생하는 중복정보를 재사용하는 배열을 사용하여, 내부 메모리의 증가량을 줄이는 방법과, 영상의 크기가 증가함에 따라 SAD 연산량도 증가함으로 계산량을 줄이고 하드웨어에 최적화 할 수 있도록 SAD 연산기의 파이프라이닝을 통하여 클록의 손실을 최대한 줄이는 방법을 제안한다.

본 논문은 다음과 같이 구성된다. 2장에서는 H.264의 인터코딩을 설명하고, 3장에서는 제안한 움직임 예측기의 각각 세부구조를 설명한다. 4장에서는 하드웨어 구현 결과를 보인다. 마지막으로 5장에서 결론을 맺는다.

Ⅱ. H.264의 인코딩

H.264/AVC 예측 부호화 기법중 하나인 화면 간 예측(inter prediction)은 움직임 추정 및 보상을 통하여 입력 영상의 시간적 중복성을 제거하는 부호화 기법이다.

일반적으로 널리 사용되는 움직임 보상을 방법은 현재 프레임의 사각형 구간 또는 '블록' 의 움직임을 보상하는 것이다. 현재 프레임에서 M x N 샘플을 갖는 블록 각각에 대해 다음의 과정이 수행된다. 현재 프레임의 M x N 샘플 블록과 일치하는 M x N샘플영역을 찾기 위해 참조 프레임의 영역을 탐색한다. 이 과정은 현재 프레임의 M x N블록과 탐색영역(일반적으로 현재 블록의 위치를 기준으로 한 영역) 내의 가능한 M x N 블록 모두 또는 일부를 비교하여, 그 중 '가장' 일치하는 영역을 찾아내는 과정이다.

현재의 M x N 블록에서 후보영역을 뺌으로써 구해지는 오차 에너지가 최소가 되는 후보 영역으로 선택하는 방법이 널리 사용된다. 가장 일치하는 영역을 찾아내는 이러한 과정을 움직임 추정(motion estimation) 이라고 한다[11].

움직임 추정은 단위블록 또는 블록와 가장 유사한 것들을 이전 프레임에서 찾아 현재 프레임의 값을 추정하는 과정이다. 움직임 추정의 목적은 되도록 많은 단위 블록 또는 블록을 이전 프레임에서 참조함으로써 시간적 중복성을 재거하여 압축률을 증가시키는데 그 목적을 둔다. 특히 H.264는 지난 표준 비디오 압축방식 과는 달리 하나 이상의 참조영상을 움직임 보상에 이용 한다. 움직임 추정 및 보상 과정에서 발생되는 움직임 정보는 각 화면 간 예측 모드별로 움직임 벡터와 참조 프레임 인덱스 정보, 그리고 해당 모드 번호로 구성된다. H.264/AVC 에서는 블록의 크기에 따라 Inter16x16, Inter16x8, Inter8x8모드, 그리고 Inter8x8 모드의 경우 8x8,8x4,4x8,4x4 모드의 서브 매크로 블록으로 구성되어 있다[12][13].

특히 다양한 가변 블록 단위의 움직임 추정을 통한 복잡하고 정밀한 움직임 영역을 표현할 수 있게 되었다. 그러나 이러한 복수 개의 참조 영상과 가변 블록은 부호화기의 움직임 예측시 최적의 움직임 벡터를 찾아내기 위한 반복적인 탐색을 요구하기 때문에 더 많은 연산량을 필요로 한다[14][15].

III. 제안한 움직임 예측기

본 장에서는 제안한 하드웨어를 구성하고 있는 각 기능 블록들의 하드웨어구조와 구현결과들을 설명한다.

1. 제안한 하드웨어 개요

1.1 제안한 H/W의 특징

고속 움직임 추정을 하기 위해서는 많은 연산량이 필요하기 때문에 움직임 추정기의 핵심 블록에 해당하는 SAD 연산기를 효율적으로 설계해야 한다. 움직임 추정을 함에 있어 외부메모리에서 현재와 참조프레임의 데이터를 많이 불러와야 한다. 외부 메모리 접근량을 줄여 SAD연산 수행시간을 단축시기기 위해서는 SAD 연산시에 중복되는 데이터를 효율적으로 읽어 들임으로써 연산속도를 높일 수 있다.

그림 1에는 최상위 블록에 해당하는 움직임 추정 프로세서의 하드웨어 구조를 나타내었다. 각각의 블록들은 버퍼, PU 어레이, SAD 선택기, MV 생성기로 구성된다. 버퍼의 구조는 현재와 참조 프레임의 데이터가 저장되어 있다. PU 어레이의 구조는 16개의 PU들이 하나로 구성되어 있고,

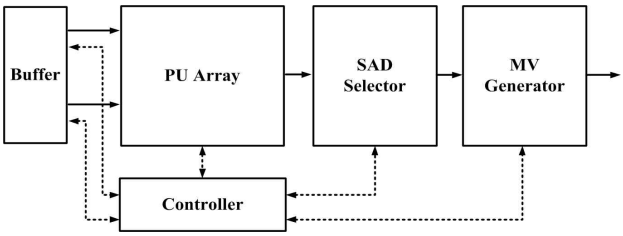


그림 1. 제안한 하드웨어의 구조
Fig. 1. Proposed architecture

PU는 16개의 PE 로 이루어져 있다. SAD 선택기는 PU 어레이에서 계산되어진 SAD 값들 중에 최소값을 선택하여 MV 생성기에서 MV 값을 출력하는 구조로 이루어져 있다. 또한 동작적인 특징은 외부메모리 접근량을 줄이기 위하여 데이터를 읽어올 때 중첩되는 데이터를 재사용하는 방법으로 계산한다. 그리고 SAD 연산을 수행할 때 클록의 손실 없이 계산할 수 있는 구조로 되어 있다.

1.2 전체적인 하드웨어 동작

제안한 하드웨어의 세부 구성요소는 외부 메모리로부터 현재 프레임 데이터와, 참조 프레임의 데이터를 저장하는 2개의 버퍼로 구성되어져 있으며, PU 어레이는 16개의 PU으로 이루어져 있으며, 여기서 가변 블록 크기의 7가지 크기에 대하여 41가지 서브 블록을 만들고, 각각의 블록들로부터 최소 SAD 값을 선택하여 움직임 벡터를 선택하여 출력하는 구조로 이루어져 있다. 제안한 움직임 추정기의

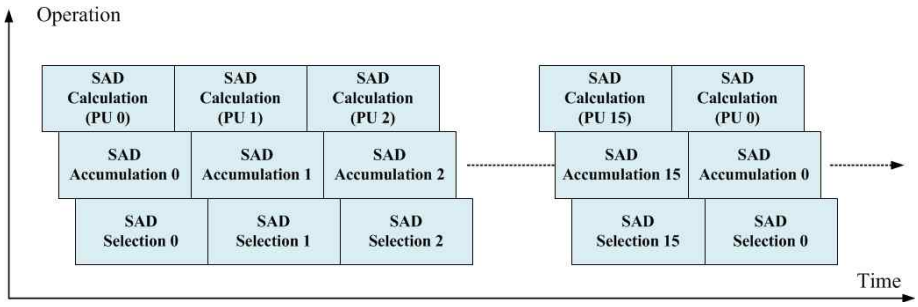


그림 2. 움직임 추정기의 파이프라인 동작
Fig. 2. Pipelining Operation of ME

전체 블록도를 그림 1에 나타내었다.

탐색범위는 64x64에 16x16 매크로블록 매칭방법으로 SAD 값을 출력한다. 현재와 참조프레임의 데이터를 각각의 PU로 입력되어 SAD 값으로 출력되어 SAD Processor를 거쳐 움직임 벡터를 찾아 가는 구조이다. 움직임 벡터 추정은 움직임의 이전의 중심벡터 값과 MVP(Motion Vector Predictor)의 차이 값이 작으면 이전의 중심에 고정이 되고, 두 값의 차이가 크면 MVP은 새로운 탐색 중심점을 찾아 가는 방법을 사용하였다[16].

PU 어레이에서 입력된 데이터들의 계산과정을 거쳐 SAD값들이 연속적으로 출력되어진다. 이러한 PU를 통하여 나온 SAD값들은 순차적으로 SAD누적기와 선택기를 통하여 최소 SAD 값들을 출력한다. 이러한 과정을 파이프라이닝 구조로 설계하였다. 위와 같은 3가지의 요소들이 계산순서에 의하여 PU0에서부터 PU15까지 하나의 PU 어레이 연산이 완료된다. 하나의 PU 어레이의 값들이 계산되어지고 나면 다음 데이터의 값들의 연산이 이루어져야 한다. 효율적으로 하드웨어를 설계하기 위하여 다음 PU어레이의 연산이 속적으로 이루어지는 구조로 되어 있어 움직임 추정의 계산을 보다 효율적으로 처리 할 수 있다. 그림 2는 움직임 추정기의 파이프라이닝 구조를 설명하고 있다.

1.3 제안한 하드웨어 동작

PE와 PU의 동작순서는 그림 3에서 설명하고 있다. PU 계산 이동시 ME 동작 방식은 PU의 계산이 진행 될 때 MB의 동작 방식 그림 3(a)에서 16x16의 입력 데이터를 각각의 16개의 PE에 순차적으로 16x1의 데이터를 총 16 Cycle 안에 처리한다. 그림 3(b) PU의 동작순서는 16개의 PE의 계산을 (0,0)을 기준으로 (0,15)까지 순차적으로 이동하면서 현재와, 참조프레임의 값을 계산하여 하나의 PU의 계산이 완료된다. 그림 4는 그림 3(b)의 PU동작순서를 좀 더 자세하게 나타내었다. 그림 4는 PU0부터 PU15까지의 동작하여 (0.63)까지 64x16의 범위를 계산한다. 이와 같은 순서로 64x64를 계산한다.

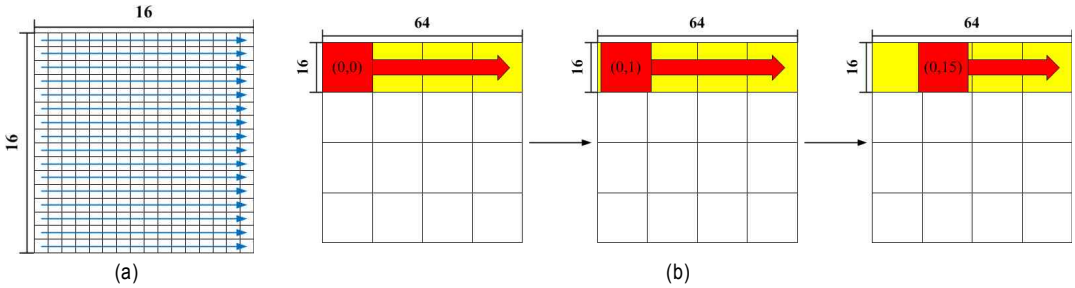


그림 3. 움직임 추정기의 동작 순서 (a) PE의 동작순서 (b) PU의 동작순서
Fig. 3. Operational sequence of (a) PE, (b) PU

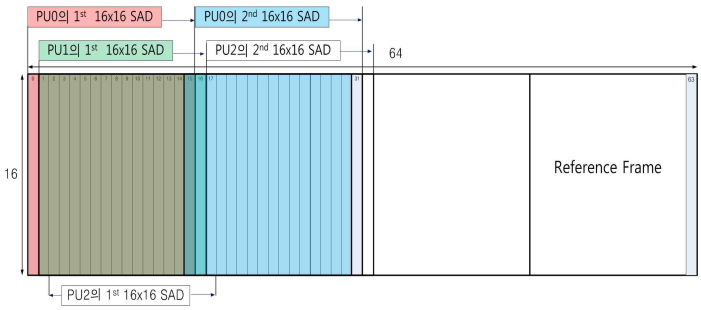


그림 4. PU의 세부동작
Fig. 4. Detail operation of PU

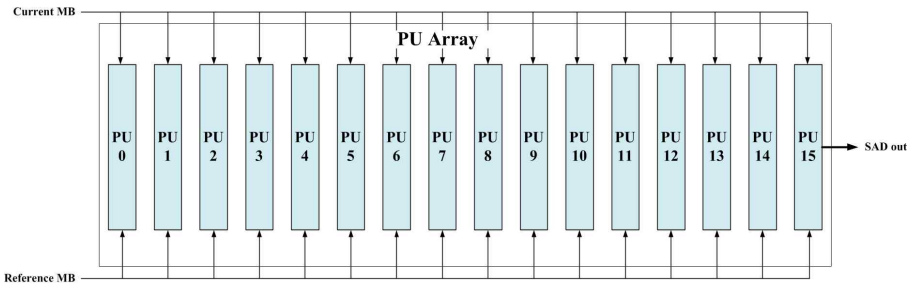


그림 5. PU 어레이의 구조
Fig. 5. Architecture of PU Arrary

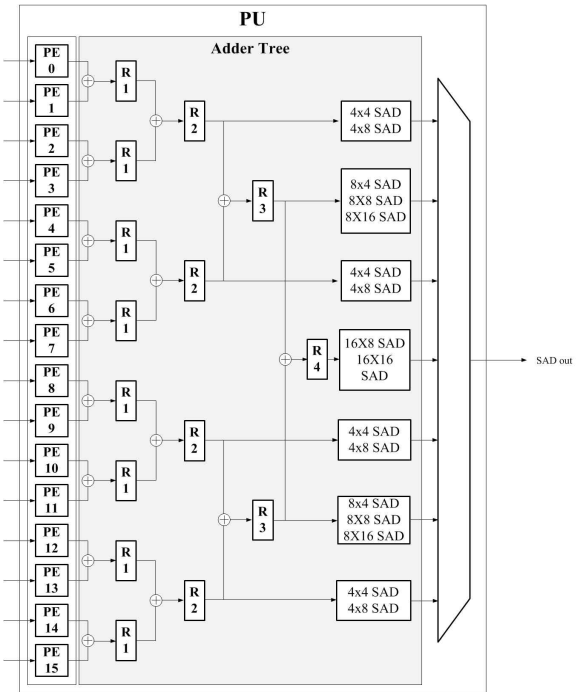


그림 6. PU의 세부구조
Fig. 6. Detail architecture of PU

2. 제안한 하드웨어 구조

2.1 Processing Unit Array 의 구조

그림 5에서 볼 수 있듯이 16개의 PU가 하나의 PU 어레이를 구성한다. 전체적인 구조는 현재와 참조 MB의 값들은 동시에 16개의 PU에 입력되어 계산되어 출력이 이루어지는 구조로 되어 있다. 입력되어진 참조 MB의 데이터는 순차적인 데이터 입력의 흐름에 따라 16개의 PU에서 계산되어지고 현재 MB 데이터는 한번 입력되면 고정되어 계산되어진다. 고정되어진 현재 MB과 데이터가 순차적으로 변하는 참조 MB의 데이터가 PU 어레이에서 연산되어 SAD 값을 출력한다.

2.2 Processing Unit 의 세부구조

하나의 PU는 16개의 PE와 15개의 레지스터가 덧셈기-트리로 구성된다. 그림 6은 PU의 구조를 나타낸다. PU는 각각의 PE로 부터 계산되어진 값들은 R1부터 R3의 레지스터에 각각 저장되고 저장된 값들은 7가지의 가변블록들의 값과 41가지의 SAD 값을 선택적으로 출력하는 구조이다.

인접한 두 개의 PE로 부터 출력된 결과를 더한 값은 R1에 저장되고, 인접한 두 개의 R1에 저장된 값을 더한 결과는 R2에 저장된다. 마찬가지 방식으로 R3와 R4에 누적된 덧셈 결과들이 저장된다. 이와 같은 연산 방법과 그림 3에서 설명한 PE의 동작을 이용함으로써 각 레지스터에 ME의 각 모드별 SAD 결과를 저장할 수 있다. 각각의 레지스터에 저장되는 모드는 그림 5에서 확인할 수 있다.

그림 7(a)는 PU가 4x4 크기의 단위로 하나의 매크로블록을 처리할 경우에 블록의 처리 순서를 나타낸다. 그림 7(b)는 클록(clk)에 따라서 PU를 구성하는 PE와 레지스터들이 어떠한 값을 출력 및 보유하는지를 나타낸다. 첫 클록에 4x4SAD 값이 출력되고, 다음 클록에 8x4,4x8 SAD값, 세 번째 클록에 8x8SAD, 네 번째 클록에 8x16, 16x8SAD 값들이 순차적으로 출력된다. 이러한 순서로 16개의 PE에서 SAD 값들이 순차적으로 블록종류에 따라 출력된다.

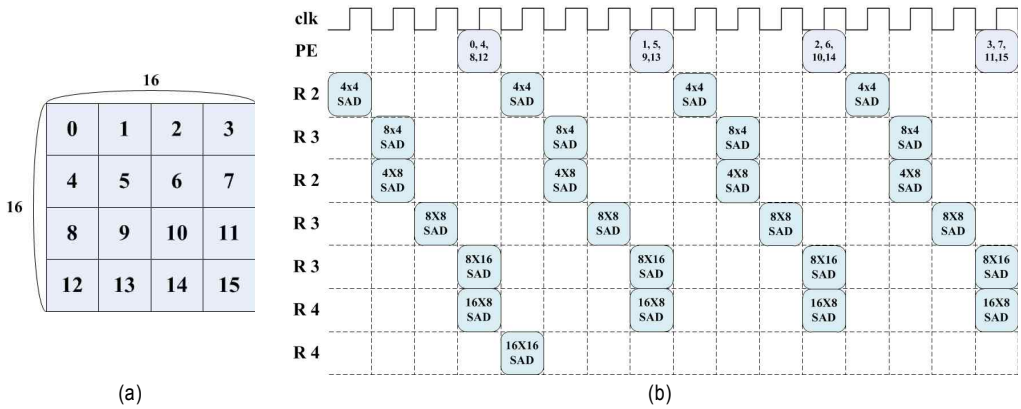


그림 7. 덧셈기-트리의 출력 사이클 (a) 16x16영역의 블록 (b) 덧셈기-트리의 클록에 따른 레지스터의 출력 값
Fig. 7. Output cycle of adder-tree (a) 16x16 block (b) register output

2.3 Processing Element의 세부구조

PE는 움직임 추정기에서 SAD 연산을 담당하는 부분이다. 하나의 PU를 이루는 16개의 PE 하나의 구조는 MB의 데이터를 이용하여 SAD값을 출력한다. SAD 연산은 차이값을 지속적으로 누적하는 연산이므로 하드웨어 구조는 절대 값, 레지스터 및 피드백 루프로 구성할 수 있다.

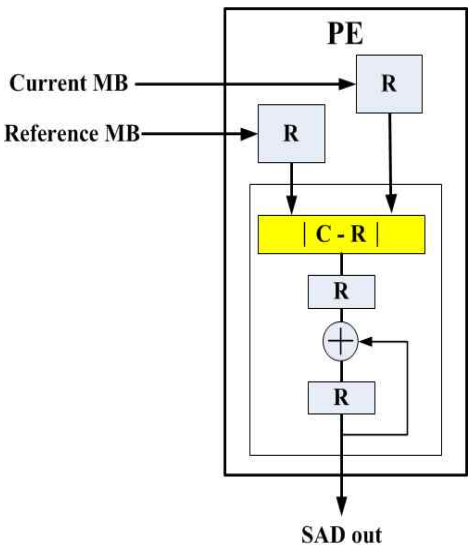


그림 8. PE의 세부구조
Fig. 8. Detailed architecture of PE.

입력 값은 그림2(a)의 순서대로 현재와 참조 MB 의 데이터를 입력받는다. PE는 현재 MB와 참조 MB의 화소값을 입력 받은 후에 두 화소간의 차이를 절대 값 형태로 구하고, 그 결과를 매크로블록 단위로 누적하는 동작을 수행한다. PE의 세부구조를 그림 8에 나타내었다.

3. On-chip 메모리

PU 어레이에서 SAD 연산을 할 때 외부 메모리 데이터 접근이 많이 이루어진다. 현재와 참조 MB의 데이터를 보다 효과적으로 사용하기 위하여 제안한 하드웨어의 구조는 외부 메모리로부터 읽어오는 데이터를 SAD 연산 시에 중복되는 부분을 재사용 하는 방법을 이용하였다. 그림 9은 이러한 데이터의 재사용 부분을 나타내었다.

그림 9(a)는 현재 MB에 적용한 데이터 버퍼링이다. Reference Frame(RF)-A 사용 시에는 RF-B에 다음 데이터를 입력시킨다. RF-B를 사용 할 때에는 RF-A에 다른 데이터를 입력시킨다. 이와 같은 방법으로 인하여 하드웨어의 량은 32x16으로 반이지만 외부 메모리로부터 접근하는 횟수는 증가 한다.

그림 9(b) 참조 MB에 적용한 데이터 버퍼링이다. 처음 64x16 (0,0)~(0.63) 탐색 범위에서 움직임 추정을 하고 나

서(1,0)~(1,63) 범위로 넘어갈 때, 이전 탐색구간과 새로운 탐색범위 사이에는 64x15 부분이 서로 공통으로 사용되는 데이터이다. 이 부분의 값을 레지스터에 저장하여 재사용되는 부분을 그대로 두고 새롭게 탐색한 부분을 외부 메모리로 부터 불러온다. 저장해둔 값과 새롭게 불러온 데이터를 가지고 (1,0)~(1,63)범위를 계산한다. 앞에서 설명한 참조 MB의 세부 구조는 그림 10에 나타내고 있다.

현재와 참조 MB에 서로 다른 데이터 버퍼링을 이용하여 움직임 추정을 할 때 현재 MB의 값들은 고정되어 있어 그림 9(a)와 같은 방법을 사용한다. 참조 MB의 데이터는 다중 참조 MB의 값들을 불러들여 계산이 이루어지므로 그림 9(b)와 같은 순서로 계산하는 것이 효과적이다. 참조MB의 데이터 버퍼링을 그림 9(a)의 방법으로 처리하게 되면 공통 부분을 다시 읽어야 하기에 처리 속도, 데이터 사용 부분에서 손실이 발생한다. 이러한 이유로 참조 MB 에 대한 데이터 버퍼링은 그림 9(b)와 같은 방법을 이용하여 데이터 재사용으로 처리 속도와, On-chip 메모리에서의 메모리 사용을 줄였다.

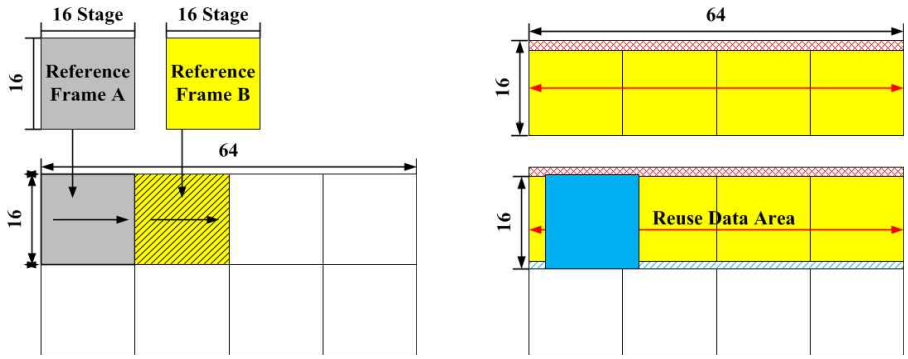


그림 9. 데이터 버퍼링 (a) 현재 MB에 적용한 데이터 버퍼링 (b) 참조 MB에 적용한 데이터 버퍼링
Fig. 9. Proposed data buffering of (a) current MB, (b) reference MB

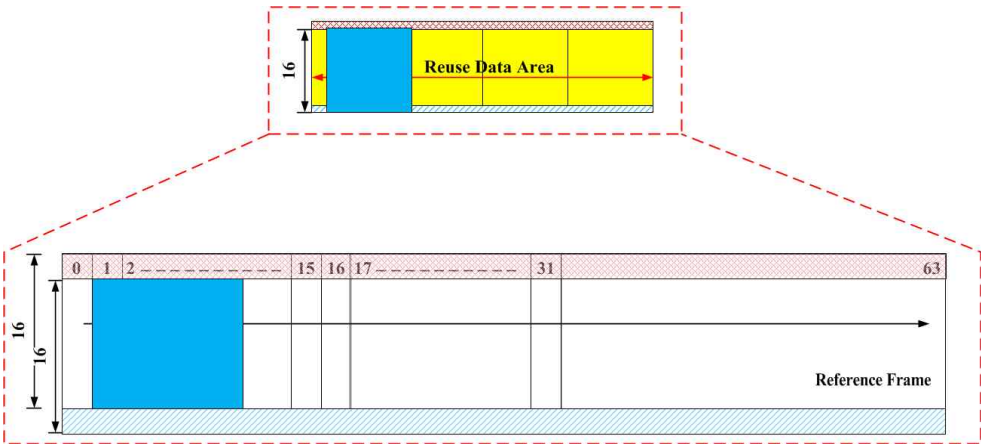


그림 10. 참조 MB에 적용한 데이터 버퍼링의 세부구조
Fig. 10. Detail structure of data buffering adapted to reference MC

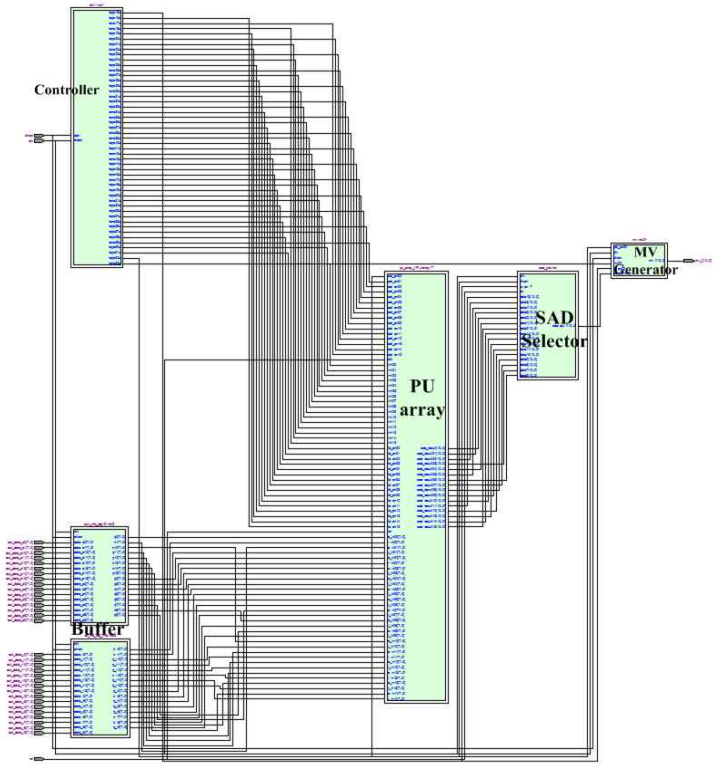


그림 11. 움직임 추정기의 RTL 합성도.
Fig. 11. RTL synthesis result of ME

Ⅳ. 하드웨어 구현 결과

본 연구는 고속 움직임 추정기를 하드웨어로 구현하기에 앞서 Verilog-HDL을 이용하여 RTL 코딩을 완성하였다. 그리고 ModelSim을 이용하여 시뮬레이션 검증을 수행하였다.

1. 구현 결과

움직임추정 하드웨어는 Altera의 Quartus Ⅱ 환경을 사용하여 구현하였다. 고속 움직임추정 하드웨어의 자원 사용률은 표 1에 나타내었고, 그림 11은 그림 1의 하드웨어를 구현한 후에 FPGA를 기반으로 하여 합성한 결과를 보여주고 있다. 또한 그림 12은 PU Array의 합성 결과를 보이고 있다.

표 1. 하드웨어 자원 사용율
Table 1. Hardware resource

Logic Utilization	Used	Available	Utilization
Combinational ALUTs	2,827	64,000	4%
Memory ALUTs	1,076	32,000	3%
Dedicated logic registers	4,523	64,000	3%
Total registers	4,523		

본 논문에서 제안한 구조로 회로를 설계하였을 경우, Altera 사의 StratixⅢ EP3SE80F1152을 타겟으로 Quartus Ⅱ 9.1을 이용하여 합성한 결과 움직임 추정부의 동작 주파수는 446.43MHz에서 안정적으로 동작함을 확인하였다. 표 1에는 구현결과 사용한 StratixⅢ EP3SE80F1152의 재원을

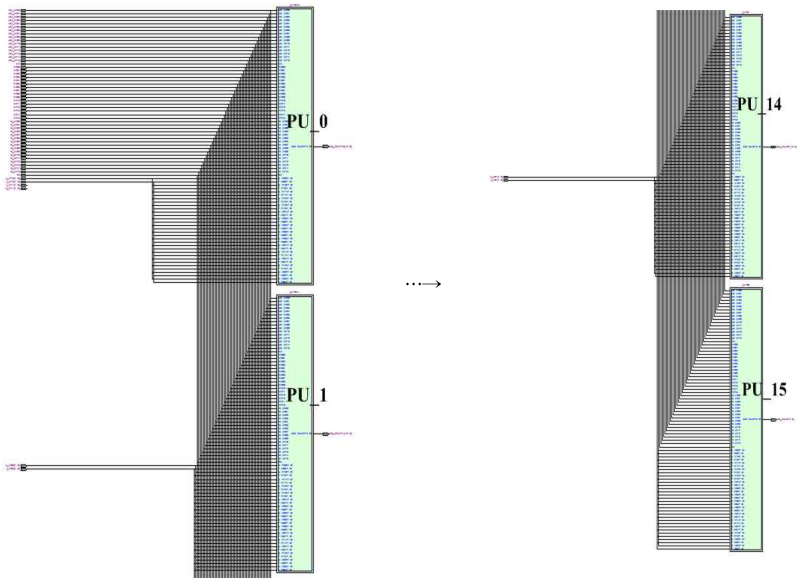


그림 12. PU 어레이의 세부 구조
Fig. 12. Synthesis result of PU Array

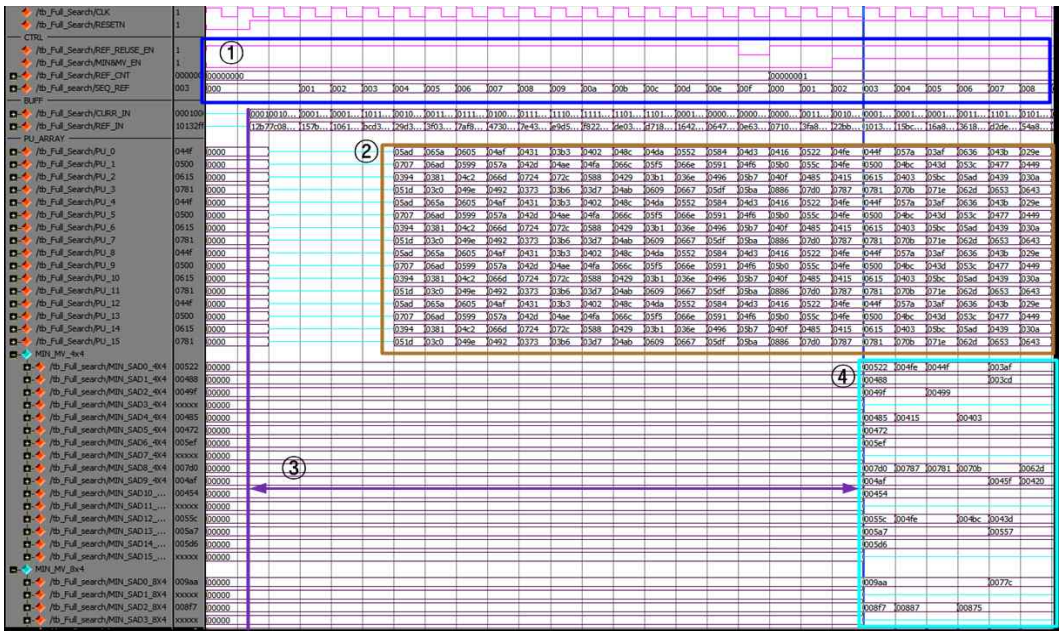


그림 13. 전체 움직임 추정기의 시뮬레이션 결과
Fig. 13. Simulation result of entire ME

나타내었다. 총 4%의 Combinational ALUT와 3%의 Memory ALUT를 사용하여 낮은 하드웨어 재원의 사용률을 보였다. 제안한 움직임 추정 하드웨어를 이용하여 1920x1080, 참조프레임의 개수 3개, 64x64 탐색범위에 16x16 단위블록을 이용하여 움직임 추정 과정을 실행한 결과 446.46MHz의 클록 주파수에서 최대 50 프레임을 처리 할 수 있었다.

2. 시뮬레이션 결과

시뮬레이션의 결과를 그림 13에 나타내었다. 그림 13은 컨트롤 신호, 데이터 입력 버퍼, PU어레이, 불록별 움직임 벡터출력 부분으로 나누어져 있다. 그림13의 ①로 표시된 부분은 컨트롤 신호들이다. REF+REUSE_EN는 신호는 16클록 마다 참조MB의 정보를 재사용하고 트리거 신호로 동작하여 참조MB을 재사용 SAD 연산을 수행하게 된다. SEQ_CNT는 전체 동작의 기준 카운터신호로 사용한다. ② 부분은 PU_0~16으로 이루어진 PU 어레이의 연산결과이다. 하나의 PU는 16개의 PE들로 이루어져 있고 각 16개의 PU별 SAD연산, Adder-tree를 통해 41개의 SAD 값을 출력한다. ③은 최초 정보 입력 이후 18클록 이후에 MIN_MV_EN 신호가 주어지면 1클록 뒤에 7가지 블록별 41개의 SAD값 중 MIN_SAD 값들이 선택되어진다. 그림 13의 ④ 부분의 신호는 선택되어진 MIN_SAD값이 19클록 번째에 움직임 벡터 값이 출력되는 것을 확인 할 수 있다.

3. 성능비교

표 2에서는 구현한 하드웨어의 성능을 비교하고 있다. [17]과 [18]의 하드웨어와 제안한 하드웨어를 표에서 보이는 바와 같이 사용 가능한 하드웨어 자원대비 실제 사용율과 동작주파수의 비교분석이 가능하다. [17]에서 제안한 하드웨어는 본 논문에서 제안한 하드웨어 보다 29.8% 이상 ALUT를 사용하였고 시뮬레이션 결과로 비교하였을 때 [17]에서 352x288 영상에서 움직임 벡터를 찾는 평균 계산 시간이 1.8ms가 소요되고 제안한 하드웨어의 시뮬레이션 결과 같은 영상을 처리하는데 0.95ms의 성능을 보여 [17]의 하드웨어보다 0.85ms 더 빠르다는 것을 알 수 있었다.

[18]의 하드웨어는 Xilinx사의 FPGA를 타겟으로 구현되었기에 본 논문의 결과와 직접 비교하기에는 어려움이 있지만, 최적의 움직임 벡터 값을 연산하는 성능에 있어서는 움직임 벡터 값을 계산하는 클록 사이클의 값을 비교해 보면 [18]에서 최적의 움직임 벡터를 찾는 데에 226 클록 사이클이 소요되고, 제안한 하드웨어는 34클록 사이클이 소요되어 약 7배 빠르게 처리 할 수 있다. 제안한 하드웨어는 메모리를 재사용하여 [18]의 하드웨어보다 총 메모리 사용 비트의 량이 약 43배 적게 사용된 것을 표2에서 확인할 수 있다.

표 2. 하드웨어 자원 사용율 비교
Table 2. Hardware resource

Logic Utilization	[17]	[18]	Proposed
FPGA	Altera Stratix II	Xilinx Virtex-5	Altera StratixIII
Total ALUTs	13,068	3,021	3,903
Total Memory bits	10,976	468,000	10,650
Suitable Motion Vector (Clk cycle)	86	226	34
Throughput	352x288 / 1.8ms	1920x1080@30fps	1920x1080@50fps
Max Frequency	100MHz	390MHz	446.43MHz

V. 결 과

본 논문에서는 고속 움직임추정기의 외부메모리 접근과, 참조 MB의 데이터를 재사용, 움직임 추정기의 동작에 대해 파이프라이닝을 하여 계산하는 고속 움직임 추정기의 구조를 제안하였고, PU의 덧셈기 트리를 이용하여 7가지의 가변블록 처리하는 하드웨어를 FPGA를 타겟으로 설계하였다. 설계한 고속움직임 추정기는 446.43MHz 주파수에서 안정적으로 동작하여 1080p 영상을 최대 50 Frame/s을 처리 할 수 있었으며, FPGA의 4%의 Combinational ALUT와 3%의 Memory ALUT만을 사용하여 낮은 하드웨어 사용률을 보였다. 고해상도 영상의 움직임을 추정하는 장치로서 많은 응용분야에서 효과적으로 사용될 것으로 기대된다.

참 고 문 헌

[1] J.-C. Tuan, T.-S. Chang, and C.-W. Jen, “On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture,” IEEE Trans. Circuits Syst. Video Technol., vol. 12, no. 1, pp. 61–72, Jan. 2002.

[2] Y. Su and M.-T. Sun, “"Fast multiple reference frame motion estimation for H.264/AVC,"" IEEE Trans. Circuits Syst. Video Technol., vol. 16, no. 3, pp. 447–-452, Mar. 2006.

[3] Y. Su and M.-T. Sun, “"Fast multiple reference frame motion estimation for H.264/AVC,"" IEEE Trans. Circuits Syst. Video Technol., pp. 447–452, Mar. 2006.

[4] Z. Liu, L. Li, Y. Song, T. Ikenaga, and S. Goto, “"VLSI oriented fast multiple reference frame motion estimation algorithm for H.264/AVC,"" in Proc. IEEE Int. Conf. Multimedia Expo, Beijing, China, pp. 1902–1905, Jul. 2007.

[5] S.Y. Yap and J.V. McCanny, "A VLSI Architecture for Variable Block Size, Video Motion Estimation", IEEE Trans. Circuits and Systems, vol.51, pp.384-389, 2004.

[6] S.Lopez, F.Tobajas, A.Villar, V. de Armas, J.F.Lopez, and R.Sarmiento,"Low cost efficient architecture for H.264 motion estimation", IEEE International Symposium on Circuits and Systems (ISCAS) Kobe, Japan, pp.412- 415, May 23-26, 2005

[7] Cao Wei, Mao Zhi Gang, "A novel VLSI Architecture for VBSME in MPEG-4 AVC/H.264"IEEE International Symposium on Circuits and Systems(ISCAS) Lobe, Japan, pp.1794 - 1797, May 23-26, 2005

[8] C.-Y.Chen, S.-Y.Chien, Y.-W.Huang, T.-C.Chen, T.-C.Wang, L.-G.Chen,"Analysis and Architecture Design of Variable Block-Size Motion Estimation for H.264/AVC", IEEE Transactions on Circuits and Systems I: Regular Papers, 16 vol.53, pp.578 - 593, 2006

[9] Chien-Min Ou, Chian-Feng Le, Wen-Jyi Hwang, "An efficient VLSI architecture for H.264 variable block size motion estimation", IEEE Transactions on Consumer Electronics, vol.51, pp.1291 - 1299, 2005

[10] J.-C. Tuan, T.-S. Chang, and C.-W. Jen, "On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture", IEEE Trans. Circuits Syst. Video Technol., Vol.12, pp. 61-72, 2002.

[11] Iain E. G. Richardson "H. 264 and MPEG-4 Video Compression" 2003

[12] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, “Overview of the H.264/AVC video coding standard”, IEEE Trans. Circuits and Systems for Video Technology, vol.13, no.7, pp. 560-576, Jul. 2003.

[13] “Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T Rec. H.264/ISO/IEC 14 496-10 AVC,"" in Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVTG050, 2003.

[14] Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification, ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, May 2003.

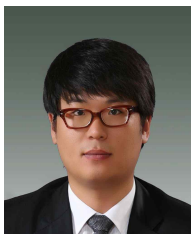
[15] J. Ostermann, T. Wedi, et al., “Video coding with H.264/AVC: tools, performance, and complexity,” IEEE Circuits and Systems Magazine, vol. 4, pp.7-28, 2004.

[16] Heejun Shim,Chong-Min Kyung, "Selective Search Area Reuse Algorithm for Low External Memory Access Motion Estimation", IEEE Transactions On Circuits And Systems For Video Technology, VOL. 19, NO. 7, July 2009

[17] Kthiri, M.; Kadionik, P.; Levi, H.; Loukil, H.; Ben Atitallah, A.; Masmoudi, N. , "Hardware implementation of fast block matching algorithm in FPGA for H.264/AVC ", Systems, Signals and Devices, 2009. SSD '09. 6th International Multi-Conference, 2009

[18] Kthiri, M.; Kadionik, P.; Levi, H.; Loukil, H.; Ben Atitallah, A.; Masmoudi, N. , "An FPGA implementation of motion estimation algorithm for H.264/AVC", I/V Communications and Mobile Network (ISVC), 2010 5th International Symposium on, 2010

저 자 소 개



임 정 훈

- 2009년 2월 수원대학교 전자공학과 졸업(공학사)
- 2009년 8월~현재 : 광운대학교 전자재료공학과 석사 과정
- 주관심분야 : 영상처리, 영상압축, SoC 설계

서 영 호

- 1999년 2월 : 광운대학교 전자재료공학과 졸업(공학사)
- 2001년 2월 : 광운대학교 일반대학원 졸업(공학석사)
- 2004년 8월 : 광운대학교 일반대학원 졸업(공학박사)
- 2003년 6월 ~ 2004년 6월 : 한국전기연구원 연구원
- 2005년 9월 ~ 2008년 2월 : 한성대학교 조교수
- 2008년 3월 ~ 2011년 2월 : 광운대학교 교양학부 조교수
- 2011년 3월 ~ 현재 : 광운대학교 교양학부 부교수
- 주관심분야 : 2D/3D 영상 및 비디오처리, 디지털 홀로그램, SoC 설계, 워터마킹/암호화

최 현 준

- 2003월 2월 : 광운대학교 전자재료공학과 졸업(공학사)
- 2005년 2월 : 광운대학교 일반대학원 졸업(공학석사)
- 2009년 2월 : 광운대학교 일반대학원 졸업(공학박사)
- 2009년 3월 ~ 2010년 2년 : 광운대학교 연구교수
- 2010년 3월 ~ 현재 : 안양대학교 정보통신공학과 조교수
- 주관심분야 : 영상압축, 워터마킹, 암호학, FPGA/ASIC 설계, Design Methodology

김 동 욱

- 1983년 2월 : 한양대학교 전자공학과(공학사)
- 1985년 2월 : 한양대학교 공학석사
- 1991년 9월 : Georgia 공과대학 전기공학과 (공학박사)
- 1992년 3월 ~ 현재 : 광운대학교 전자재료공학과 정교수 신기술 연구소 연구원
- 2000년 3월 ~ 2001년 12월 : 인터스닷컴(주) 연구원
- 2009년 3월 ~ 현재 : 광운대학교 실감미디어연구소 연구소장
- 2006년 3월 ~ 현재 : (사) 실감미디어산업협회 이사
- 주관심분야 : 3D 영상처리, 디지털 홀로그램, 디지털 VLSI Testability, VLSI CAD, DSP 설계, Wireless Communication