H.264/AVC용 가변 블록 크기를 지원하는 움직임 추정 부호기의 연구

김원삼* · 손승일*

A Study on Motion Estimation Encoder Supporting Variable Block Size for H.264/AVC

Won-sam Kim* · Seung-il Sonh*

이 논문은 2008년도 한신대학교 연구비 지원에 의해 수행되었음.

요 약

인터 예측의 핵심 요소는 ME와 MC이다. ME는 SAD(Sum of Absolute Difference)와 같은 정합기준을 사용하는 것뿐만 아니라 비트스트림의 최종 비트수에 따라서 최적의 움직임 벡터를 찾는다. 인터 예측 부호화는 고화질의 실시간 비디오 응용에 있어서 언제나 주된 병목을 초래한다. 따라서 실시간 비디오 응용에서는 인터 예측을 수행하는 고속의 전용 하드웨어를 필요로 한다. 본 논문에서는 H.264/AVC의 움직임 추정기를 연구하였다. 설계된 움직임 추정기는 2-D 시스토릭 배열 기반으로 기본 처리기 요소를 병렬로 연결하여 SAD 값을 빠르게 계산한다. 참조데이터를 상위영역과 하위영역으로 나누어 각각의 연결선을 두고 입력 시퀀스를 조절하여 파이프라인 중지 없이 연속적인 연산을 수행한다. 데이터 재사용 기법을 통하여 메모리 엑세스를 줄였고 특별한 지연 없이 최소의 SAD를 갖는 파티션을 찾아내어 움직임 벡터를 생성하게 하였다. 설계된 움직임 추정기는 가변 블록 크기를 지원하며 하나의 매크로블록의 연산을 하는데 328 사이클이 소요된다. 논문 [6]이 로컬메모리를 사용하는 것과 달리, 본 논문은 로컬 메모리를 사용하지 않는다.

ABSTRACT

The key elements of inter prediction are motion estimation(ME) and motion compensation(MC). Motion estimation is to find the optimum motion vectors, not only by using a distance criteria like the SAD, but also by taking into account the resulting number of 비트s in the 비트 stream. Motion compensation is compensate for movement of blocks of current frame. Inter-prediction Encoding is always the main bottleneck in high-quality streaming applications. Therefore, in real-time streaming applications, dedicated hardware for executing Inter-prediction is required. In this paper, we studied a motion estimator(ME) for H.264/AVC. The designed motion estimator is based on 2-D systolic array and it connects processing elements for fast SAD(Sum of Absolute Difference) calculation in parallel. By providing different path for the upper and lower region of each reference data and adjusting the input sequence, consecutive calculation for motion estimation is executed without pipeline stall. With data reuse technique, it reduces memory access, and there is no extra delay for finding optimal partitions and motion vectors. The motion estimator supports variable-block size and takes 328 cycles for macro-block calculation. The proposed architecture is local memory-free different from paper [6] using local memory. This motion estimation encoder can be applicable to real-time video processing.

키워드

Video Coding, H.264/AVC standard, Variable Block Size Motion Estimation

* 한신대학교 정보통신학과

접수일자 2008. 05. 16

Ⅰ. 서 론

움직임 추정(ME : Motion Estimation)은 비디오 신호의 시간방향 중복성분을 효과적으로 제거함에 따라, 비디오 부호화 시스템에서 널리 채용되고 있는 핵심적인 기술이다. 비록 움직임추정 기술의 연구가 오랫동안 지속되어 왔지만 보다 정확하고 빠른 움직임 벡터의 탐색에 대한 연구는 현재까지도 그 요구가 지속되고 있다[1].

이 중 부분 탐색 블록정합 알고리즘은 가능성 없는 후보 블록 여부를 판별하기 위한 계산이 증가하는 단점을 지닐뿐더러 부분 후보 블록이 제거되었을 때 불규칙적인 데이터 흐름으로 연산 사이클이 늘어나게 되고 이를 하드웨어로 구현할 경우 입력 제어가 복잡하며 실제 연산 사이클도 늘어나게 되는 문제점을 가진다[2].

이에 반해 알고리즘 상으로 연산수가 많은 완전탐색 블록정합 알고리즘은 탐색영역 내의 모든 지점에 존재하는 후보블록에 대한 정합 연산으로부터 최적의 정합 블록을 추출함으로써 압축정보 비트수를 감소시킬 수 있다.

Ⅱ. 관련 연구

논문 [5]는 1D PE 배열을 이용한 구조이며 적은 수인 16개의 PE로 높은 Throughput을 달성하였다. 모든 가변 블록 크기를 지원하며 탐색 영역의 크기인 p=8일 때 하나의 매크로블록 당 4096 사이클이 필요로 한다. 이것은 전통적인 1D PE 배열에 비해 높은 Throughput과 가변블록크기를 지원하는 장점을 갖지만 1D PE 배열의 한계로 인하여 인코더에서는 움직임 추정 연산으로 인한 병목현상이 여전히 존재한다.

반면에 논문[6]은 1D PE 배열을 병렬적으로 구성하여 2D PE 배열로 구현되었다고 볼 수 있으며 4×4 블록 단위의 모듈 16개가 동시적으로 SAD 연산을 수행한다.

마찬가지로 모든 블록 크기를 지원하고 p=8 일 때 하나의 매크로블록 당 256 사이클 내에 블록 정합 연산을 마칠 수 있는 구조를 갖는다. 그러나 이 구조는 수직 방향의 17개의 화소로 구성된 4개의 열이 동시적으로 PE 내부의 로컬 램에 입력되어야한다. 따라서 움직임 추정 연산을 하기 전에 참조 영상이 미리 버퍼링이 되어야 하기 때문에 이에 해당하는 저장 공간이 추가적으로 필요하다.

이에 본 논문은 2D PE 배열의 장점을 이용하고 논문 [6]의 단점을 해결하기 위한 움직임 추정의 아키텍처를 연구하였다.

Ⅲ. 제안하는 H.264/AVC 움직임 추정 모듈

본 논문에서는 완전탐색 블록 정합 알고리즘인 SAD (Sum of Absolute Difference)를 정합 척도로 사용하고 2-D 시스토릭 배열(Systolic Array)을 이용한 움직임 추정기를 제안한다.

시스토릭 어레이 방식을 이용하여 구현할 경우 단순한 데이터 흐름에 의해 제어 회로가 간단해지며 2-D 시스토릭 배열에 의해 고속 연산을 수행 할 수 있는 장점을 가지고 있을 뿐만 아니라 시스토릭 어레이의 처리기 요소(PE : Processing Element)에서 발생되는 부분합(PS : Partial Sum)으로 H.264/AVC의 가변블록크기의 블록 정합을 16×16크기의 연산만으로도 쉽게 구할 수 있다[4].

제안된 움직임 추정기의 구조는 그림 1과 같다. 전체 시스템은 주소발생기(Address generator), 전치 배열(Transpose Array), 쉬프트 레지스터(Shift register array), SAD 병합기(SAD merger), 비교기(Comparator), 움직임 벡터 예측기(Motion vector predictor) 및 제어기(Controller)로 구성되어 있다.

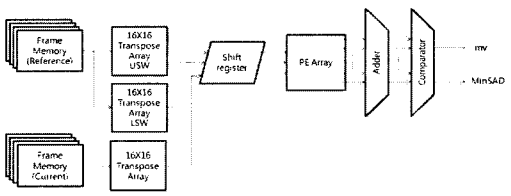

그림 1. 제안하는 움직임 추정 모듈 구조
Fig. 1 Structure of the proposed ME module

먼저 주소 발생기에서 필요한 화소의 주소를 호출하여 현재 프레임 메모리와 참조 프레임 메모리에서 전치 배열에 입력한다. 전치 배열은 PE 어레이의 SAD연산을 위해 수직방향의 화소들을 수평방향으로 바꿔주며 매 사이클 마다 USW, LSW, 현재 블록의 각각 1개의 화소가 쉬프트 레지스터를 통과하여 PE Array에 입력되고 SAD 연산에 사용된다. 변형된 노드 연결에 의해 4×4블록 단

위로 SAD값이 출력되면 SAD 병합기에서 H.264/AVC에서 지원되는 모든 가변 블록 사이즈의 SAD를 계산하게 된다. 이후 비교기에서는 최소 왜곡을 갖는 정합 블록을 찾아내어 움직임 벡터를 생성하게 된다. 다음으로는 각 기능 블록에 대해 자세히 설명하고자 한다.

가) 전치 배열

주소발생기에 의해 메모리부터 화소데이터들이 4 화소씩 행단위로 입력된다. 이후에 설명될 PE 배열 구조는 한 행 또는 한 열씩 SAD를 계산하는 구조로 이루어져 있는데, 한 열씩 처리하는 것이 한 행씩 처리하는 것 보다 메모리 접근량에 있어서 효율적이다. 왜냐하면 매크로 블록의 처리 순서는 왼쪽으로부터 오른쪽 열 방향으로 처리하도록 규정되어 있는데, 이전에 호출한 화소 데이터의 재사용을 위해 매크로블록의 탐색범위에서 오른쪽 16×32에(p=8일 때) 해당하는 블록이 다음 매크로블록 인터 예측 시 탐색범위의 왼쪽 탐색범위가 됨으로 오른쪽 탐색범위만을 호출하기 때문에 중복되는 메모리 접근을 줄일 수 있다. 따라서 입력된 현재 매크로블록 및 탐색 범위를 전치시켜주어야 한다. 전치 배열은 Bin Sheng[3]이 제안한 전치 배열 구조를 움직임 추정에 적용하기 위하여 수정하였다. 16×16 매크로블록 단위의 처리가 가능하게 하고, PE 배열 구조의 입력 순서에 맞출 수 있도록 수정하였으며 제안하는 구조는 그림 2와 같다.

주소발생기에 의해 입력되어진 4개의 화소는 전치배열 입력으로 들어가며 Demux는 16개의 화소를 4화소씩 나누어 차례대로 입력하며 16개의 화소가 16회 전부 입력되면 스위치가 토글되어 입력 방향을 바꾼다.

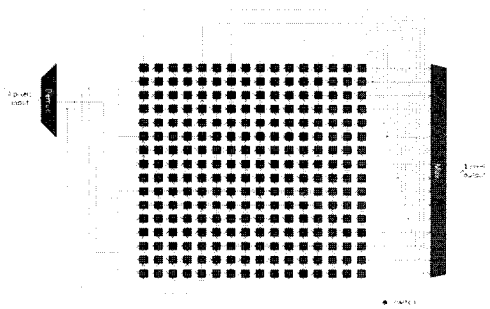

그림 2. 제안하는 16×16 전치 배열
Fig. 2 Proposed 16x16 transpose array

이때 출력부는 스위치에 의해 입력된 방향과 다른 방향으로 출력되어지고 PE 배열 입력에 맞추어 한 사이클당 하나의 화소씩 Mux에 의해 출력되어진다. 따라서, 입력에는 4 화소 입력 단위로 64 사이클이 소요되고, 출력에는 한 화소씩 256 사이클이 소요된다.

이 16×16 전치 배열 3개를 이용하여 각각 현재 프레임의 화소, 참조 프레임의 USW(Upper Search Window) 화소, 참조 프레임의 LSW(Lower Search Window) 화소에 사용한다.

나) 처리기 요소

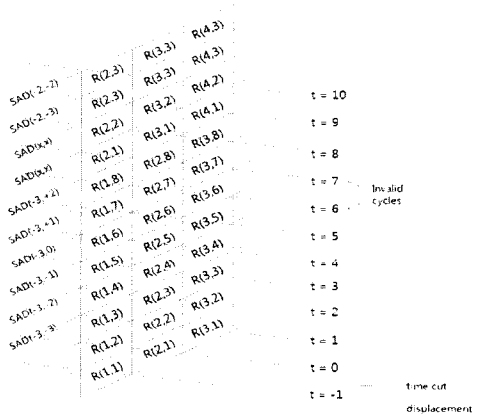

그림 3. 차기 수평 블록왜곡 연산에서의 불연속 연산
Fig. 3 Discontinuous operation in the next horizontal block distortion operation

그림 3-10은 단순한 입력 신호 흐름에 의해 참조 블록 데이터가 PE 배열에 입력되면 차기 수평 블록왜곡 연산에서 불연속 연산이 일어나는 경우를 나타내고 있다. 즉, t=1에서 움직임 벡터 (-3,-3)에 위치하는 후보블록의 첫 행과 기준 블록의 첫 행 사이의 차이 값 절대치 연산이 이루어진다.

다음 사이클 t=2에서는 (-3,-2)위치하는 연산이 이루어지고 t=3에서는 (-3,-3)에 있는 후보블록의 마지막 행의 차이의 절대값 연산이 이루어져 처음으로 완전한 후보블록 왜곡 연산이 이루어진다. 그런데 t=7에서는 다음 수평 변위의 후보블록의 연산을 할 경우 연산에 필요한 해당 데이터가 준비되지 않아 불필요한 연산을 수행하게 된다. 2 사이클(블록의 너비에서 1을 뺀 크기)의 무효한 연산시간 이후 (-2,-3)에 위치한 후보블록의 첫 행과 기준블록의 첫 행간의 차이의 절대값 연산이 이루어진

다. 이러한 불필요한 연산은 매 기준블록마다 발생하며 수평 탐색 범위 크기만큼 발생하게 된다. 이러한 무효 사이클을 제거하기 위해 그림 4와 같이 탐색 영역 데이터를 세 영역으로 구분하여 입력하는 방식이 널리 사용된다[4].

그림 4에서 보듯 탐색 영역은 세 영역으로 구분한다. 탐색영역의 최상단 첫열에서 N번째 열까지를 상단영역(UR : upper region), 최하단 열에서 상향으로 N개의 열을 포함하는 영역을 하단영역(LR : lower regin), 두 영역에 속하지 않는 나머지 영역을 중간영역(MR : middle region)으로 구분한다. 상단과 중간영역의 데이터는 USW입력선을 따라 그림 4(b)에 나타난 화살표 방향으로 1, 2, ..., 2ph+N의 순서로 시스토릭 어레이에 순차적으로 입력시킨다.

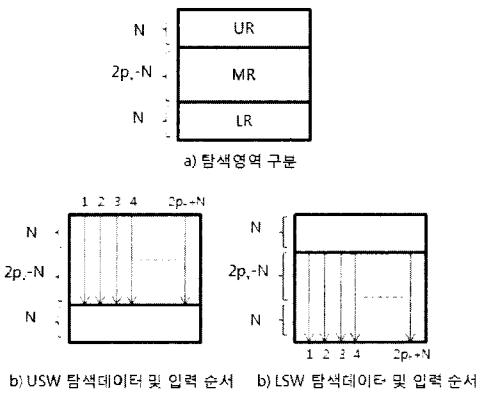

그림 4. 탐색영역 구분과 데이터 입력
Fig. 4 Search area division and input data

또한 중간과 하단영역의 탐색 데이타는 LSW 입력선을 따라 그림 4(c)에 나타난 화살표 방향으로 1, 2, ..., 2ph+N의 순서로 순차적으로 어레이에 입력시킨다. 이러한 입력 시퀀스는 최좌단에서 오른쪽 행 방향으로 수평 탐색 변위를 한 화소씩 증가시키면서, 또한 수평 변위에서는 최상단의 수직 탐색변위에서 한 화소씩 증가하는 방향으로 블록처리를 수행하게 된다.

하지만 제안하는 움직임 예측은 N = 16이고 수직 탐색 범위 Pv = 8이기 때문에 2Pv-N은 0이므로 중간영역은 존재하지 않으며 왼쪽의 PE에서 우측의 PE로 부분 합이 전달 될 때의 지연만을 고려해 하나의 수직방향 USW 탐색데이터 입력 후 1사이클 dummy 데이터를 입력해주고 dummy 데이터가 입력되는 즉시 수직방향으로 LSW 탐색데이터가 입력되면 이웃하는 차기 기준블록의 블록처리를 지연 없이 연속으로 처리할 수 있다.

USW와 LSW로 나뉜 32×32 탐색범위는 다시 2개의 16×32 블록으로 나뉜다. 이는 PE에 별도의 레지스터를 이용하여 차기 기준블록에 사용되는 중첩된 탐색 영역 데이터를 충분히 저장할 수 있기 때문이다. 그림 5는 두 기준 블록 A와 B의 탐색 데이터 영역을 나타낸다.

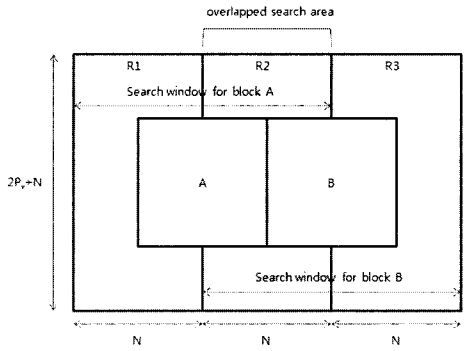

그림 5. 기준 블록에 따른 탐색 영역 겹침
Fig. 5 Search area overlap according to reference block

현재 블록 A에서 탐색영역 R2는 차기 블록 B에서도 사용된다. 따라서 A에서 PE 배열에 R1을 입력하여 수직방향으로 SAD 연산을 수행하기 시작하여 R2를 연속적으로 입력받으면 지연 없이 연산이 가능하고 R2의 데이터가 PE 배열에서 연산 초기 R1의 위치에 오게 되었을 때 각 PE내부의 레지스터에 저장을 한 뒤 블록 B의 탐색을 시작할 때 불러오면 R3에 대한 데이터만 엑세스 하게 되어 불필요한 메모리 엑세스 없이 연산이 가능하다.

그림 6은 USW와 LSW의 입력선을 추가한 처리기 요소의 내부를 나타낸다. PE는 현재 블록 데이터와 참조 블록 데이터의 AD를 계산한다. 계산된 AD는 이웃한 PE(i-1,j)로부터 전달된 부분 합을 합하여 이웃한 PE(i+1,j)에게 전달한다. 여기서 LC, LU, LL은 각각 현재 블록 데이터, 상위영역 데이터, 하위영역 데이터의 래치(Latch)를 의미한다. 이 래치는 스탠딩 래치(Standing Latch)와 실행 래치(Execution Latch)로 나뉘며 스탠딩 래치는 현재 프레임의 다음 매크로블록의 움직임 추정 연산 시 재 사용될 화소를 저장해 놓는 래치이고, 실행 래치는 현재 프레임의 현재 매크로블록 움직임 예측 연산을 위해서 AD에 입력 될 화소가 저장된 래치이다. 또한

Lm은 제어부로부터 입력되어 USW 혹은 LSW의 데이터를 선택하는 제어신호를 저장하는 래치이고 다음 사이클에 PE(i+1,j)로 한 사이클 지연되어 전달된다. 또한 각 PE는 2단계의 파이프라인 기법을 적용하였으며 이를 위해 L1과 L2 래치가 사용된다. 제안하는 PE 구조를 조합한 16×16 PE 배열은 그림 7과 같다.

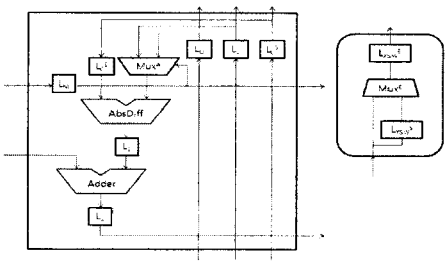

그림 6. PE의 내부구조
Fig. 6 Internal structure of PE

여기서 초기 입력에 쉬프트 레지스터 31개가 존재하는 이유는 dummy 데이터로 인해 256개의 화소가 입력되어지는 사이클은 16 사이클이 증가하였고 LSW의 입력은 USW보다 15사이클 뒤늦게 입력되므로 입력에 있어 총 31사이클이 더 소요되기 때문이다.

16×16 PE 배열에서 4, 8, 12, 16 열의 PE는 부분 합을 SAD 병합기로 전달한다. 이는 4×4 블록 단위의 SAD를 구하면 H.264/AVC에서 지원되는 모든 가변 블록 사이즈의 SAD를 구할 수 있기 때문이다. 그림 8은 PE 배열의 부분합의 경로를 보여주고 있다. 회색의 PE는 입력되는 부분 합이 0인 경우이고 나머지 PE는 부분 합이 이웃한 PE(i-1,j)에서 전달되는 경우를 나타낸다.

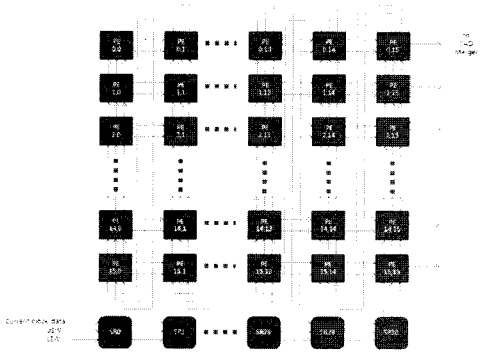

그림 7. 제안하는 PE 배열
Fig. 7 Proposed PE array

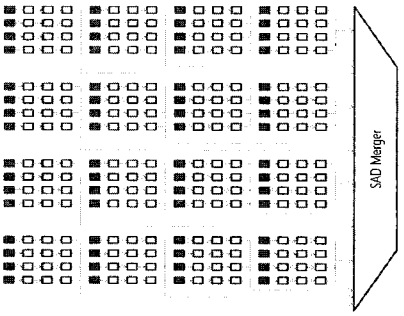

그림 8. PE 배열의 부분합 경로
Fig. 8 Partial sum path of PE array

다) SAD 병합기(SAD merger)

PE 배열의 출력은 4×4 블록내에 4×1 블록의 부분 합이다. H.264/AVC의 가변블록 사이즈의 최소 크기는 4×4 블록이므로 각각의 4×4블록 내의 4개의 행의 부분 합을 더해주면 16개의 4×4 블록에 대한 SAD를 얻게 된다. 그림 9는 4×1 블록의 SAD 64개가 더해져 4×4블록의 SAD 16개가 출력되어 레지스터에 저장된다. 저장된 SAD는 다음 4×8 블록과 8×4 블록의 SAD 16개를 구하는데 사용되며 동시에 비교기에 입력된다.

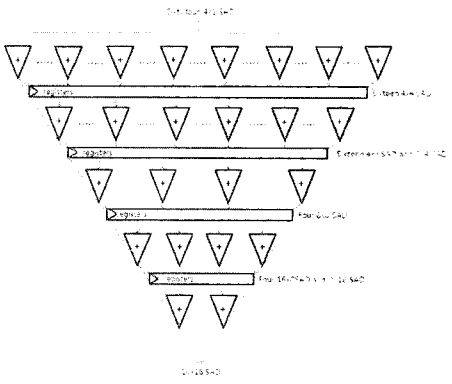

그림 9 SAD 병합기의 구조
Fig. 9 Structure of SAD merger

마찬가지로 4×8, 8×4, 8×8, 8×16, 16×8, 16×16 블록에 대한 SAD도 구해지고 비교기로 출력된다. 모든 가변 크기의 블록은 289개의 벡터 중 하나가 선택 될 수 있기 때문에 첫 SAD 출력이 나간 후 연속적으로 각각의 블록 크기마다 탐색 범위내의 벡터에 따른 나머지 288개의 SAD도 연속적으로 출력된다.

라) 비교기(Comparator)

SAD 병합기에서 출력된 각각의 블록 크기의 SAD는 벡터별 289개가 순차적으로 비교기에 입력된다. 비교기는 두 가지의 기능을 한다. 첫 번째로는 각각의 블록 크기별로 최소의 SAD를 갖는 벡터를 구한다. 두 번째로는 매크로블록 내에서 가능한 모든 블록 조합 중 최소의 SAD를 갖는 파티션을 찾아낸다.

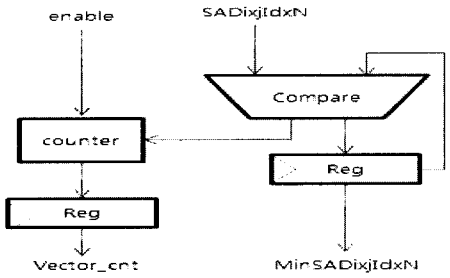

그림 10 각 블록별 최소 왜곡 출기
Fig. 10 Minimum distortion detector for each block

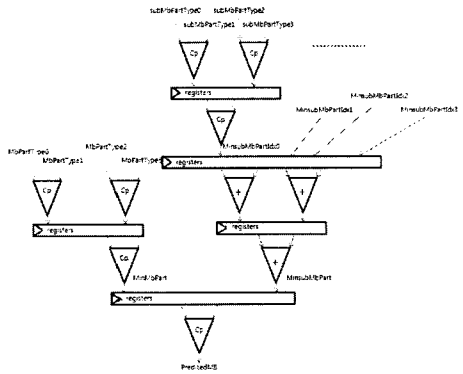

그림 11. 매크로블록의 최적의 파티션을 찾는 검출기의 구조
Fig. 11 Structure of the detector finding optimal partition for MB

그림 10은 첫 번째 기능인 각각의 블록별 최소 SAD를 갖는 벡터와 최소의 SAD를 찾아내는 구조를 보여준다. 이 모듈은 289사이클 동안 입력되어지는 개개의 블록의 SAD를 이전에 입력된 SAD와 비교하여 최소의 SAD 값을 구한다. enable신호가 인가되면 카운터가 동작하며 SAD 비교 연산 중 작은 SAD입력이 들어 왔을 때 카운트를 레지스터에 신호를 전달하여 최종적으로 최소의 SAD에 해당되는 카운트를 출력하게 된다. SAD 병합기의 출력은 16개의 4×4 블록, 8개의 4×8 블록, 8개의 8×4 블록, 4개의 8×8 블록, 2개의 8×16 블록, 2개의 16×8블록 그리고 16×16 블록의 SAD이므로 총 41개의 블록별 최소 왜곡 비교기 모듈이 필요하다.

각 블록별 최소 왜곡 검출기로부터 매크로블록 파티션 4개의 조합과 8×8 블록 당 서브매크로블록 파티션 4개씩 16개의 조합을 생성하여 최소의 SAD를 갖는 파티션을 찾아내야한다. 그 후 최적의 매크로블록 파티션과 서브매크로블록 파티션을 비교하여 최소의 블록 왜곡을 갖는 파티션과 각각의 파티션의 벡터를 얻게 된다.

Ⅳ. 구현 및 성능비교

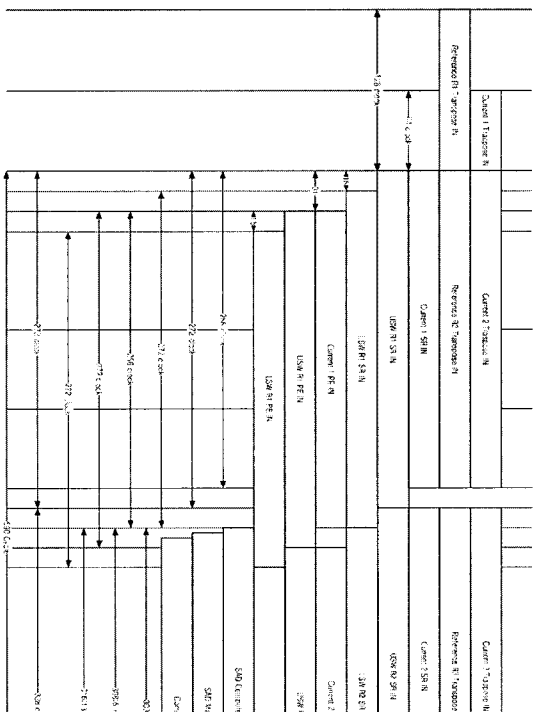

그림 12. 제안하는 움직임 추정기의 타이밍
Fig. 12 Timing for the proposed motion estimator

지금까지 제안하는 움직임 추정기의 각 모듈별 동작 및 구조를 설명하였다. 그림 12는 움직임 추정기의 타이밍을 보여준다. 첫 128 사이클은 그림 13의 현재 블록 A와 탐색 범위 R1을 읽어오는 상태이다. 현재 블록 256화소는 32비트 대역폭 시스템에서 한 사이클 당 4화소씩 읽어오기 때문에 64 사이클이 소요되고 USW와 LSW 또한 각각 64 사이클 씩 128 사이클이 소요된다. 다음 상태로는 참조 메모리로부터 전치 배열에 입력 된 블록 A와 탐색 범위 R1이 출력되어 쉬프트 레지스터를 통하여 PE 배열에 입력되고 동시에 제어부에 의해서 블록 B와 탐색범위 R2가 전치배열에 32비트의 여유가 생길 때마다 입력된다.

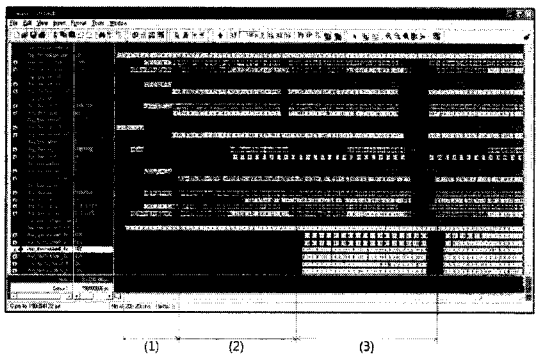

그림 13. 움직임 추정기의 시뮬레이션 결과
Fig. 13 Simulation result of motion estimator

즉 4 사이클 당 1 번 엑세스하며 총 256사이클에 걸쳐 입력된다. 탐색 범위의 데이터들은 16번의 유효한 사이클과 한 번의 무효한 dummy 입력 사이클이 존재하기 때문에 이 두 번째 상태는 272 사이클이 소요된다. 세 번째 상태는 현재 블록 C와 탐색 범위 R2를 읽어오는 상태이며 PE 배열에서 탐색 범위 R0의 데이터가 쉬프트를 끝내고 SAD 계산이 시작되는 단계이다. 또한 출력이 나가기 시작하면 SAD 병합과 비교 연산까지 탐색 블록 당 328 사이클이 반복적으로 소요된다. 한 행의 매크로블록이 모두 처리 되었을 때 다시 첫 번째 상태로 되돌아가 위에 설명한 절차를 반복하게 된다. 그림 3-22는 마찬가지로 움직임 추정기의 시뮬레이션 결과를 보여주고 있다. 타이밍과 마찬가지로 (1), (2), (3)은 현재 블록 A, B, C의 입력 시점이 되고 (3)에서 첫 번째 연산 결과가 나오기 시작함을 알 수 있다.

완전 탐색 블록정합 알고리즘인 SAD와 2-D 시스토릭 배열을 이용하여 움직임 추정기를 구현하였다. 제안된 움직임 추정기는 탐색 영역의 입력을 2개로 나누어 시스토릭 배열 기반에서 지연없이 고속처리가 가능하도록 구현하였다.

다음 표 1은 이전에 제안된 시스토릭 배열 기반의 움직임 추정기와의 성능 비교를 나타내었다. 하나의 매크로블록을 처리하는데 필요한 사이클 수를 T, 생성되는 최적의 움직임 벡터의 수를 M이라고 했을 때 Throughput S는 다음과 같이 정의된다.

$$S = \frac{M}{T} \qquad (식\ 1)$$

논문[5]에서는 1D PE 배열을 이용했기 때문에 논문[6]과 본 논문에 비해 낮은 면적과 매크로블록 당 많은 사이클이 필요로 한다. 따라서 Throughput은 가장 낮고 이는 Low-end에 적합하다. 논문[6]은 2D PE 배열을 이용하여 가장 높은 Throughput을 갖는다. 하지만 수직 방향 17개의 화소 열 4개가 동시적으로 입력되어야하기 때문에 매우 넓은 대역폭이 필요로 하고 이를 해결하기 위해 움직임 추정 연산 모듈 전단에 추가적인 내부 프레임 버퍼를 두었다.

표 1. 움직임 추정기의 성능 비교
Table 1. Performance comparison for motion estimators

	논문 [5]	논문 [6]	본 논문
PE개수	16	256	256
사이클/MB	4096	256	328
Throughput (S)	41/4096	41/256	41/328
가변 블록 크기지원	○	○	○
로컬메모리	X	○	X
특이사항	- 1-D 배열로 추정시간이 김	- 내부 로컬 메모리 필요 - 넓은 대역폭을 요구함	- 간단한 메모리 인터페이스 가능

이에 반해 본 논문은 16x16 크기의 전치 배열 3개를 이용하여 32비트 대역폭 시스템에 적용가능하며 PE 내부에 16x16 스탠딩 래치 3개를 통해 데이터 재사용 기법을 실현하여 메모리 엑세스 빈도를 크게 감소시켰다.

V. 결론

본 논문은 H.264/AVC의 가변 블록 크기를 지원하는 움직임 추정기를 제안하였다. 움직임 추정은 인코더의 연산량에 60~70% 차지할 만큼 복잡도가 높기 때문에 [7] 실시간 처리를 위한 하드웨어 구현이 필요하다. 제안하는 구조는 2D PE 배열을 사용하여 높은 Throughput을 달성하였고 부분 합의 데이터 경로만을 변경하여 특별한 하드웨어의 증가 없이 H.264/AVC의 모든 가변 블록 크기를 지원한다. 기존의 대역폭 문제 또는 높은 메모리 사용 문제를 해결하였으며 참조된 데이터의 재사용 하여 메모리 액세스의 빈도수를 줄였기 때문에 High-end에 적합한 구조라 예상된다. 또한 제안하는 구조에서 사용된 전치배열과 데이터 재사용 기법은 하드웨어 면적에 제한이 큰 응용 분야에도 적용 가능할 것이다.

참고문헌

[1] 강현수, 박성모, "다중 참조 영상 움직임 추정을 위한 고속 전역탐색법," 전자공학회 논문지 제43권 SP편 제1호, pp.1~8 2006. 1.

[2] 안태경, 문용호, 김재호, "적응적 다단계 연속 제거 알고리즘," 한국통신학회논문지, 제29권 제1C호, pp.111~118, 2004. 1.

[3] Bin Sheng, Wen Goo and Di Wu, "An implemented architecture of deblocking filter for H.264/AVC," 2004 International Conference on Image Pfocessing (ICIP)

[4] 남승현, "동영상 압축부호화를 위한 고속 이동벡터 추출기 VLSI설계," 연세대학교 박사학위논문, 1995. 12.

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

[6] Chien-Min Ou, Chian-Feng Le and Wen-Jyi Hwang, "An Efficient VLSI Architecture for H.264 Variable Block Size Motion Estimation," IEEE Transactions on Consumer Electronics, Vol.51, No.4, Nov. 2005.

[7] Peter Kuhn, "Algorithms, complexity analysis and VSLI architectures for MPEG-4 motion estimation," Kluwer Academic Publishers, 2003.

저자소개

김원삼(Won-Sam Kim)

2008년 한신대학교 컴퓨터정보학과(석사)

※관심분야 : ASIC 설계, H.264/AVC 코덱

손승일(Seung-Il Sonh)

1989년 연세대학교 전자공학과 (공학사)
1991년 연세대학교 대학원 전자공학과(공학석사)
1998년 연세대학교 대학원 전자 공학과(공학박사)
2002년~현재 한신대학교 정보통신학과 부교수

※관심분야 : ATM 통신 및 보안, ASIC 설계, 영상신호처리칩, 비디오코덱