논문 99-24-11A-19 한국통신학회논문지 '99-11 Vol.24 No.11A

범용 움직임 추정 프로세서 설계

준회원 서 영 산*, 정회원 유 재 희*

Design of a General purpose Motion Estimation processor

Young-san Seo*, Jae-hee You* *Regular Members*

요 약

다양한 움직임 추정 알고리즘의 연산이 가능한 범용 움직임 추정 프로세서 아키텍처가 제안되었다. 제안된 새로운 아키텍처는 여러 영상 압축 표준 응용 분야와 MPEG 4에 있어 객체별 최적부호화 등에 활용 가능하다. 제안된 아키텍처는 프로그램 가능한 새로운 I/O 인터페이스 방안을 통해 전역탐색 블록 정합 알고리즘 및 여러 가지 계층 탐색 알고리즘 연산이 가능하도록 가변성을 갖는다. 알고리즘간의 성능 비교와 단일 연산 유닛을 사용한 설계 방안, 프로세서의 VLSI 구현방안이 서술되었다. 실제영상을 바탕으로 C++를 이용하여, 주요 하드웨어 모듈에 대하여 VHDL을 이용해 검증이 수행되었다.

ABSTRACT

A general purpose motion estimation processor architecture is presented. The presented architecture is capable of computing multiple motion estimation algorithms for various image compression standard application areas and the content based coding in MPEG 4. Full search block matching and various hierarchical search algorithms can be adaptively computed using programmable I/O interfaces. The performance comparisons among motion estimation algorithms and the architecture based on identical PE as well as VLSI implementation methodologies are described. The verifications using C++ and VHDL are discussed for real image data and major hardware blocks respectively.

I. 서 론

멀티미디어 화상처리는 다양한 알고리즘 연산이 필요하며, 일반적으로 저화질의 경우이거나 연산량이 적은 알고리즘에 있어서는 범용 신호처리 프로세서를 사용한다. 그러나 움직임 추정 (ME : Motion Estimation)은 방대한 연산량을 필요로 하여 ASIC구현이 필수적이다. 움직임 추정 알고리즘은 일반적으로 정확도가 좋고 안정된 성능을 보이는 블록 정합 알고리즘 (BMA : Block Matching Algorithm)을 이용하여 구현된다. 그리고 BMA에는 움직임 벡터를 구하는 방법에 따라 크게 나누어 전역 탐색 블록 정합 알고리즘 (FBMA : Full Search Block Matching Algorithm)과 계층 탐색 알고리즘 (HSA : Hierarchical Search Algorithm)이 있다. FBMA는 비교적 정확한 움직임 추정을 수행할 수 있으나 탐색 영역이 넓어질 경우 필요한 연산량이 크게 증가하는 문제점이 있다. 따라서 탐색 영역이 증가함에 따라 HSA가 FBMA에 비해 바람직하나, 지역 최소값에 빠질 위험을 해결할 필요가 있다.

기존의 FBMA 아키텍처를 살펴보면 2차원 시스톨릭 어레이를 사용하는 경우[1], 데이터 입력에 필요한 포트의 개수가 많고 PE (Processing Element)의 개수가 대상 MB (Macro Block) 크기로 제한되므로 응용 분야에 따라 연산 능력 가변성이 떨어진다. 또한 [2]의 경우 하드웨어의 이용도가 100%이

* 홍익대학교 전자공학과(jaeheeu@wow.hongik.ac.kr)
논문번호 : 99117-0328, 접수일자 : 1999년 3월 28일
※ 본 연구는 한국 과학재단 (과제번호 : KOSEF 961-0919-102-1)의 지원을 받아 수행되었습니다.

고 입력 데이터의 중복 사용도를 높일 수 있으나 탐색 영역의 크기가 사용되는 MB 크기의 절반에 해당하는 특수한 경우로 제한된다. 1차원 시스톨릭 어레이 구조[3][4]를 사용하는 경우 2차원 어레이에 비해 입출력에 사용되는 포트 수는 감소 하나 하드웨어의 이용도가 떨어진다. HSA 아키텍처에 있어서는 파이프라인 방식의 트리 아키텍처[5]를 이용하여 세 단계 HSA를 구현하였으며, 트리 컷을 사용하였다. 그러나 PE 트리의 말단 부분에 일시에 많은 과거 프레임 데이터를 공급해야 하고, 파이프라인 인터리빙을 사용하여 MB 출력을 위한 입·출력 대역폭이 크게 증가한다. 따라서 PE에 데이터를 효율적으로 공급할 수 있는 아키텍처[6]가 제안되었으나 구현상 복잡하다는 단점이 있다.

본 논문에서는 위와 같은 문제점을 해결하고 FBMA, HSA 등 여러 알고리즘의 효율적인 실시간 처리를 위한 범용 움직임 추정 프로세서 아키텍처가 제안되었다. II장에서는 FBMA 및 HSA 움직임 추정 알고리즘을 소개하고, 이들의 장·단점을 분석하였다. III장에서는 상위 아키텍처 설계 방안이 소개된다. IV장에서는 중요 모듈에 대한 VLSI 구현 방안이 제시되었다. V장에서는 성능 분석을, VI장에서는 C++을 이용하여 실제 화상을 바탕으로한 검증 결과와 설계된 하드웨어에 관한 VHDL 검증 결과가 논의되었다.

II. 움직임 추정 알고리즘

BMA는 움직임 벡터를 구하는 방법에 따라 크게 FBMA와 HSA로 나눌 수 있다. FBMA는 탐색 영역내의 모든 MB와 비교하여 이중 차이가 최소인 값을 구하고, 이를 바탕으로 두 MB간의 위치 변화를 움직임 벡터로 발생시키는 방법이다. 탐색 영역이 넓으면 넓을수록 보다 큰 움직임에 대해 추정이 가능하다.

식(1)은 FBMA의 알고리즘을 나타낸다. 식(1)에서 PF는 이전 프레임을, CF는 현재 프레임을 나타낸다.

$$EV_{FBMA}(i,j)=\sum_{y=0}^{N-1}\sum_{x=0}^{N-1}abs(PF(x+i,y+j)-CF(x,y)) \quad (-p \le i, j < p)$$

Block size : N × N, 탐색 영역 : p (1)

EV(i, j) : error value between PF and CF

FBMA는 정확도는 높으나 HDTV와 같이 탐색 영역이 넓어질 경우 비교해야할 화소수가 막대하게 증가하여 하드웨어의 양이 크게 증가하는 단점이 있다. 따라서 연산량을 감소시키기 위하여 HSA에서는 단계별로 MB단위 또는 MB내의 화소단위에 대해 샘플링 정도를 조정하며 연산을 수행한다. 첫 단계에서는 넓은 탐색 영역에 대해 큰 샘플링 거리로 정확도가 떨어지는 움직임 벡터를 찾고, 단계가 진행됨에 따라 상위 움직임 벡터를 바탕으로 설정된 작은 탐색 영역에 대해 보다 정확도가 높은 움직임 벡터를 찾는다. 따라서 탐색 영역이 증가하여도 상대적으로 구현이 용이하다는 장점이 있다.

$$EV_{HSA1}(i,j)=\sum_{y=0}^{\frac{N}{S_{st}}-1}\sum_{x=0}^{\frac{N}{S_{st}}-1}abs(PF(x\times S_{st}+i,y\times S_{st}+j)-CF(x\times S_{st},y\times S_{st})) \quad (2)$$

S_{st} : 스테이지 샘플링 거리 $(-p_{st} \le i, j < p_{st})$

화소 샘플링 계층 탐색 알고리즘(HSA1)은 탐색 범위내의 모든 PF MB를 탐색의 대상에 포함시키는 대신 식 (2)에 나타낸 $EV_{HSA1}(i, j)$와 같이 PF와 CF의 각 MB에서 단계별로 화소 샘플링 거리를 변화 시켜가며 절대값 차이를 구한 후 각 MB단위로 모두 더하여 구하게 된다. $EV_{HSA1}(i, j)$에서 i, j는 PF와 CF MB의 위치를 나타내게 되며 탐색 영역내의 전 (i, j) 쌍에 대하여 계산하게 된다.

MB 샘플링 계층 탐색 알고리즘 (HSA2)은 화소 샘플링대신 MB 단위 샘플링을 행하는 것이 차이점이다. HSA1, HSA2에 있어서 각각 샘플링에 의해 연산에서 제외되는 화소와 MB가 있기 때문에 상위 단계에서 잘못된 움직임 벡터를 찾을 경우 지역 최소값을 산출할 가능성이 있다.

HSA3는 지역 최소값 문제점을 완화시키기 위하여 샘플링 거리내의 화소에 대하여 LPF를 사용하여 화소의 평균값을 구하여 이를 연산의 대상으로 함으로써 정확도를 향상시키는 방안이다. 따라서 다른 두 알고리즘에 비하여 지역 최소값에 빠질 염려가 적으나 LPF를 위한 하드웨어가 추가로 필요하다.

표 1에 탐색 영역 크기 32, 360 × 240의 크기를 갖는 football, telephone, train 등의 화상에 대하여 각 알고리즘의 PSNR (Peak to peak Signal to Noise Ratio), MSE (Mean Square Error), 연산량의 비교 결과를 나타내었다. 표 1에서는 움직임의 성질에 따라 수치가 가장 높은 것을 1로, 가장 낮은

것을 4로 하여 각 알고리즘의 순위를 표기하였다.

종합적으로 살펴보면 FBMA는 가장 정확도가 높고 규칙적인 연산구조로 이루어져 하드웨어의 구현이 비교적 용이하다는 장점이 있으나, 실시간 처리 시 필요 하드웨어의 양이 막대하다는 단점이 존재한다. HSA는 연산량이 상대적으로 적어 SDTV-525 영상 규격이나 HDTV급 영상 규격 등의 넓은 탐색 영역에 적합하나 정확도가 떨어지는 단점이 있다.

표 1. 움직임 추정 알고리즘의 연산량 및 성능 비교

	움직임 방향이 일정한 화상		불규칙한 움직임을 갖는 화상		연산량 순위
	MSE	PSNR	MSE	PSNR	
FBMA	4	1	4	1	1
HSA1	2	4	1	3	2
HSA2	1	3	2	4	2
HSA3	3	2	3	2	2

HSA1의 경우에는 비디오폰과 같이 움직임의 방향이 일정한 경우에 적합하며, 움직임이 불규칙적인 경우 HSA2를 사용하는 것이 좋다. 움직임이 느릴 경우에는 FBMA, HSA3대신 HSA1 또는 HSA2를 사용하는 것이 좋다. HSA3은 적은 연산량에 비하여 상당히 우수한 성능을 나타낸다. 특히 MPEG 4에서는 객체별로 부호화를 수행할 필요가 있으므로 부호화 시 객체별 특징에 따라 다양한 움직임 추정 알고리즘을 사용하여 최적화 시킬 수 있다. 이와 같이 각각의 움직임 추정 알고리즘은 정확도, 연산량, VLSI 구현 용이도 등에서 장·단점이 있으므로 응용분야에 적합한 알고리즘을 바탕으로 연산할 필요성이 있다.

Ⅲ. 범용 움직임 추정 프로세서 설계방안

그림 1에 FBMA, HSA등의 다양한 알고리즘의 연산이 가능한 제안된 움직임 추정 프로세서의 전체 아키텍처를 나타내었다.

1. I/O 인터페이스

I/O 인터페이스에 해당하는 화상 데이터 버퍼, LPF와 MUX는 범용성을 위하여 여러 알고리즘에서 요구되는 다양한 입력 패턴의 처리를 위해 프로그램 가능하도록 설계되었다. 여기서 LPF는 HSA3의 경우 화소의 평균값을 구한다.

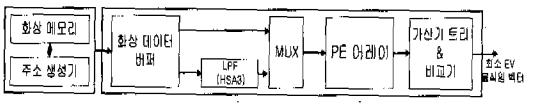


그림 1. 움직임 추정 프로세서 전체 아키텍처

표 2. 화상 데이터 입력 구조

FBMA	HSA1	HSA2	HSA3
(0, 0)	(0, 0)	(0, 0)	$[(0, 0) + (1, 0) \cdots (S_{st} - 1, S_{st} - 1)]/S_{st}^2$
(1, 0)	(S_{st}, 0)	(S_{st}, 0)	$[(S_{st}, 0) + (S_{st}+1, 0) \cdots (2S_{st} - 1, S_{st} - 1)]/S_{st}^2$
(2, 0)	($2S_{st}$, 0)	($2S_{st}$, 0)	$[(2S_{st}, 0) + (2S_{st}+1, 0) \cdots (3S_{st} - 1, S_{st} - 1)]/S_{st}^2$
(3, 0)	($3S_{st}$, 0)	($3S_{st}$, 0)	$[(3S_{st}, 0) + (3S_{st}+1, 0) \cdots (3S_{st} - 1, S_{st} - 1)]/S_{st}^2$
…	…	…	…

표 2에 I/O 인터페이스 설계를 위하여 각 알고리즘의 입력 데이터 구조를 나타내었다. FBMA에서는 MB내의 연속된 화소를 입력시키고, HSA 연산의 경우 각각 MB내의 샘플링된 화소 또는 선택된 MB내의 연속된 화소를 필요로 한다. 따라서 이와 같은 각 경우를 프로그램 가능하게 하기 위하여, 또한 HSA의 경우에는 각 움직임 추정 계층에 적합하게 화상 데이터가 입력되도록 하기 위하여 각 계층의 샘플링 거리(S_{st})의 데이터 간격을 두고 입력되도록 한다. 이를 위해 $(S_{st})^2$로 나눈 나머지별로 화상 데이터가 저장된 화상 메모리를 사용하여 I/O 및 메모리 access rate를 최소화시킨다. 그림 1에 나타낸 주소 생성기에서는 FBMA의 경우 샘플링 없이, HSA에 있어서는 표 2에 나타낸 바와 같이 샘플링 거리에 따라 화상 메모리 주소를 생성하여 적절한 화상 데이터를 외부 화상 메모리로부터 연산부로 입력시킨다. 그리고, HSA3의 경우 HSA1을 바탕으로 샘플링된 화소 대신 LPF를 거쳐 화소 평균값을 MUX를 통해 연산부로 입력함으로써 FBMA, HSA 1, 2, 3 모두를 동일한 인터페이스로 처리할 수 있게 하였다.

2. 단일 연산 유닛 구성 방법

여러 가지 움직임 추정 알고리즘에서 화소간의 절대값 차이를 구하고, 이를 축적한 후 가장 작은 차이값을 찾아내는 연산은 동일하므로 이를 공통적인 하드웨어로 구현하였다. 또한 다양한 움직임 추

정 알고리즘을 동일한 아키텍처로 연산하기 위해서는 알고리즘마다 혹은 HSA의 각 계층에 있어서 다른 연산량의 처리에 대한 고려가 필요하다.

그 방법을 그림 2와 식 (3)에 나타내었는데, 계층이 증가함에 따라 샘플링 거리 및 탐색 영역이 감소함으로 인하여 발생되는 연산량의 증가와 탐색 영역의 감소로 인하여 발생되는 연산량의 감소 비율을 동일하게 함으로써 연산량과 I/O rate를 계층에 무관하게 하여 FBMA와 더불어 HSA 모두 각 계층마다 동일한 하드웨어로 연산 가능하게 하였다.

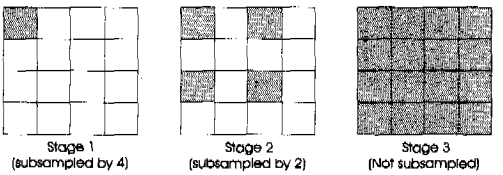


그림 2. 계층별 I/O rate와 연산량과의 관계

$$\left(2p_{st}\times\frac{N}{S_{st}}\right)^2=\left(2p\times 2^{-n}\times\frac{N}{2^{2-n}}\right)^2=\frac{(pN)^2}{4} \quad (3)$$

HSA 스테이지 0 : n = 0 (subsample by 4)
스테이지 1 : n = 1 (subsample by 2)
스테이지 2 : n = 2 (subsample by 1)
스테이지 3 : n = 3 (subsample by 1)

또한 HSA 계층 마다 EV 계산 시 가산되어야 할 화소수와 비교되어야 할 EV 수는 변하나 합은 일정하다. 이를 위하여 비교기는 가산기를 변형시켜 구현하여 통일적 아키텍처를 유지하도록 하였다.

3. PE 어레이

절대값 차이를 연산하는 PE의 설계는 움직임 추정 연산 성능에 가장 큰 영향을 미치게 된다. 제안된 아키텍처에서는 움직임 추정연산의 실시간 처리를 위하여 다양한 움직임 추정 알고리즘 내에 공통적으로 존재하는 병렬성을 최대한 이용하였다. 즉, 식 (2)의 $x(x\times S_{st}+i)$, $y(y\times S_{st}+j)$의 화상 데이터에 대한 연산에 있어 각 PE가 i, j에 대해 병렬 연산을 수행하도록 설계되었다. PE는 실시간 처리를 위하여 어레이 형태로 구현하였으며 필드 및 프레임 연산이 가능하도록 필드 단위로 연산을 수행하여 프레임 연산의 경우에는 이를 조합하여 결과를 산출할 수 있도록 하였다.

PE는 그림 3에 나타낸 바와 같이 TFPU (top field processing unit)와 BFPU (bottom field processing unit)로 구성된다. 설계된 PE는 알고리즘에 적합하게 샘플링된 화소 또는 연속된 화소를 입력받아 절대값 차이를 계산하게 된다. PE는 2 x 2 화소 크기의 화소 그룹을 연산하도록 하였으며, TFPU, BFPU는 각각 2 x 1 화소 크기를 연산하게 된다. PE의 클록에 따른 연산 구조는 다음과 같다. 먼저 연산의 첫 번째 클록에서 PE의 SR(Shift register)에 일시에 그림 1에 나타낸 화상 데이터 버퍼로부터 화상 데이터를 전달한다. PE는 매 클록마다 CF 데이터와 PF 데이터와의 절대값 차이를 구하여 그림 1의 비교기/가산기 트리에 전달한다. 연산이 종료된 PF 데이터는 재사용 되도록 SR을 통하여 다른 PE로 이동되며, 다음 화소 데이터 행에 대한 움직임 추정 연산 시에 필요한 데이터는 해당 PE행 밑에 인접해 있는 PE에 연결된 SR로부터 공급받게 된다.

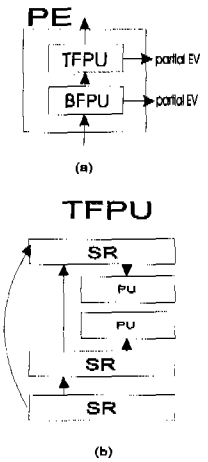


그림 3. PE의 구성
(a)PE의 구조 (b)기본 연산 유닛

4. 계산 속도와 하드웨어 양 조절 방안

MPEG 등에서 규정하는 응용분야에 따른 여러 가지 영상 규격을 동일한 하드웨어로 처리하기 위해서는 하드웨어 양의 조절이나 연산 속도를 조절할 수 있어야 한다. 제안된 아키텍처에서는 PE의 각 PU와 연결되는 SR의 수를 증가시켜 인터리빙을 통해 요구되는 연산 하드웨어의 양을 줄일 수 있으며, 연산 능력을 확장시키기 위해서는 PE의 수를

증가시킴으로써 가능하다. 또한 탐색 영역을 확장하기 위해서는 단지 SR 수만을 늘림으로써 구현할 수 있다.

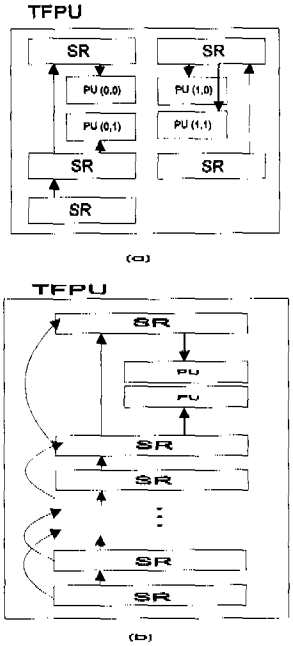


그림 4. 연산 능력 및 탐색 영역 조절 방안
(a) 연산 능력 확장 (b) 탐색 영역 확장

그림 4는 연산능력과 탐색 영역을 확장하기 위한 방안을 보여주고 있다. 그림 4 (a)에서 나타낸 바와 같이 기본 TFPU를 여러 개 배열함으로써 동시에 여러 개의 화소의 병렬 처리가 가능하며, 기본 TFPU를 바탕으로 단지 SR의 개수를 증가시켜 동일한 PU 개수로 탐색 영역의 확장이 가능하다.

5. Timing diagram

표 3에 N = 16, p = 8일 때 4개의 PE(총 16개의 PU)를 사용하여 HSA 1에 대한 연산에 있어서의 PE의 연산 과정 예를 나타내었다. PE내의 PU배치는 그림 4(a)의 경우를 가정하였고 4개의 PE로 확장된 형태이다. 동시에 4개의 4×4 화소 그룹을 계산할 수 있으며 표 3에서 각 시간마다 4개 PU의 내부 PE들의 동작을 나타내었다. HSA 1이므로 CF, PF 데이터는 4의 거리를 두고 샘플링된 데이터를 연산하며 PE(0, 0)을 기준으로 PE(1, 0), PE(0, 1), PE(1, 1)은 각각 수평으로 1 오른쪽, 수직으로 1 아래쪽, 수평 1 오른쪽, 수직 1 아래쪽의 거리를 갖는 화소를 연산하게 된다. 표 3에서도 나타낸 바와 같이 CF 데이터는 PE내에서 공통으로 다수 클록동안 사용되므로 global 버스를 사용하여 보다 느린 속도로 PE에 공급하도록 하여 VLSI 구현을 용이하게 하였다.

6. 하드웨어 효용도 향상 방안

제안된 아키텍처에서는 HSA 연산에 있어, 상위 계층의 움직임 벡터에 해당되는 화소 데이터를 PE 어레이에 제한된 입력포트 수를 통해 입력시킴으로써 발생되는 CF MB 연산간의 지연 시간을 제거해 100%의 하드웨어의 이용도를 갖도록 하였다.

표 3. PE의 연산 과정 예

Time	PU	PE(0, 0)	PE(1, 0)
0	(0, 0)	|CF(0, 0) - PF(0, 0)|	|CF(1, 0) - PF(1, 0)|
	(1, 0)	|CF(4, 0)- PF(4, 0)|	|CF(5, 0) - PF(5, 0)|
	(0, 1)	|CF(0, 4) - PF(0, 4)|	|CF(1, 4) - PF(1, 4)|
	(1, 1)	|CF(4, 4) - PF(4, 4)|	|CF(5, 4) - PF(5, 4)|
1	(0, 0)	|CF(0, 0) - PF(4, 0)|	|CF(1, 0) - PF(5, 0)|
	(1, 0)	|CF(4, 0) - PF(8, 0)|	|CF(5, 0) - PF(9, 0)|
	(0, 1)	|CF(0, 4) - PF(4, 4)|	|CF(1, 4) - PF(5, 4)|
	(1, 1)	|CF(4, 4) - PF(8, 4)|	|CF(5, 4) - PF(9, 4)|
		⋮	
63	(0, 0)	|CF(8, 8) - PF(20, 20)|	|CF(9, 8) - PF(21, 20)|
	(1, 0)	|CF(12, 8) - PF(24, 20)|	|CF(13, 8) - PF(25, 20)|
	(0, 1)	|CF(8, 12) - PF(20, 24)|	|CF(9, 12) - PF(21, 24)|
	(1, 1)	|CF(12, 12) - PF(24, 24)|	|CF(13, 12) - PF(25, 24)|

Time	PU	PE(0, 1)	PE(1, 1)
0	(0, 0)	|CF(0, 1) - PF(0, 1)|	|CF(1, 1) - PF(1, 1)|
	(1, 0)	|CF(4, 1) - PF(4, 1)|	|CF(5, 1) - PF(5, 1)|
	(0, 1)	|CF(0, 5) - PF(0, 5)|	|CF(1, 5) - PF(1, 5)|
	(1, 1)	|CF(4, 5) - PF(4, 5)|	|CF(5, 5) - PF(5, 5)|
1	(0, 0)	|CF(0, 1) - PF(4, 1)|	|CF(1, 1) - PF(5, 1)|
	(1, 0)	|CF(4, 1) - PF(8, 1)|	|CF(5, 1) - PF(9, 1)|
	(0, 1)	|CF(0, 5) - PF(4, 5)|	|CF(1, 5) - PF(5, 5)|
	(1, 1)	|CF(4, 5) - PF(8, 5)|	|CF(5, 5) - PF(9, 5)|
		⋮	
63	(0, 0)	|CF(8, 9) - PF(20, 21)|	|CF(9, 9) - PF(21, 21)|
	(1, 0)	|CF(12, 9) - PF(24, 21)|	|CF(13, 9) - PF(25, 21)|
	(0, 1)	|CF(8, 13) - PF(20, 25)|	|CF(9, 13) - PF(21, 25)|
	(1, 1)	|CF(12, 13) - PF(24, 25)|	|CF(13, 13) - PF(25, 25)|

즉 움직임 추정 연산을 수행하는 동안 다음 CF MB 연산에 쓰일 PF 데이터를 움직임 추정 프로세서 외부로부터 입력받아 화상 데이터 버퍼에 저장한 후 현재의 CF MB에 대한 연산이 끝나자마자 PE 어레이내의 SR로 일시에 전달하여 CF MB 연산 사이에 PF 데이터를 받아들이기 위한 지연시간을 제거하였다. 그리고 이후 연산에 필요한 데이터

는 외부로부터 순차적으로 입력시켜 사용하게 된다. 상기와 같은 화상 데이터 버퍼와 SR 구조를 사용하여 다음 연산 전에 순차적으로 일시에 필요한 데이터를 나누어 입력함으로써 I/O rate 및 포트 수를 최소화시킬 수 있다.

Ⅳ. 모듈별 VLSI 구현 방안

움직임 추정 연산을 위해 사용되는 공통 하드웨어인 PE와 비교기/가산기 트리의 VLSI 구현 방안을 제시하였고, 이에 대해 VHDL로 검증하였으며, 회로 합성 및 PE에 대한 레이아웃이 수행되었다.

PE는 연산 속도를 결정하는 부분이므로 이에 대한 고려가 필요하다. 설계된 절대값 연산기는 그림 5에서 나타낸 것과 같이 AECU(Absolute error calculation unit)에서 (PF - CF) 연산이 수행되고 음수인 경우 2의 보수를 출력하게 된다. 이 경우 2의 보수 생성 하드웨어를 최적화 시킴으로써 연산 속도를 향상 시켰다.

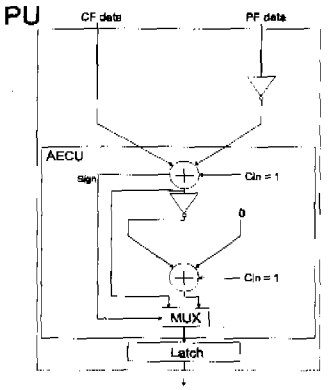


그림 5. Processing unit의 구조

움직임 추정 연산에 필요한 PF 데이터는 매 클록마다 SR로부터 공급받게 되고, CF 데이터는 global 버스를 이용하여 공급받는다. 그림 6에 PU와 SR이 결합된 핵심 연산유닛의 레이아웃 결과를 나타내었으며 PU와 SR의 간격은 배선 지연시간 감소를 위하여 최대한 가깝게 배치하였다. LG CMOS 0.6um 공정을 바탕으로 161.4 X 517.8um^2 의 크기를 갖는다.

비교기/가산기 트리에 있어서는 하나의 하드웨어를 비교기와 가산기로 선택적으로 사용할 수 있는 구조로 설계하였다. 이를 통하여 다양한 탐색 영역

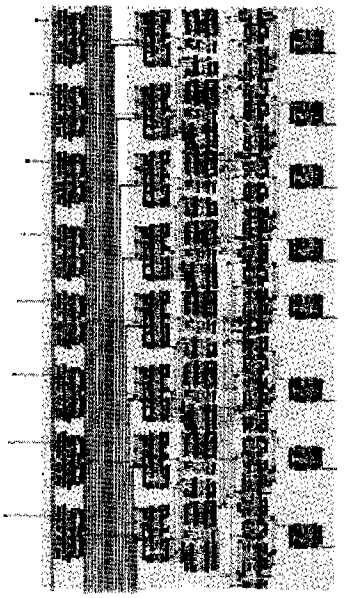


그림 6. PE 어레이 레이아웃

및 샘플링 거리에 따라 통일적 아키텍처로 구현 가능하다. 2 x 2 크기를 갖는 화소에 대해 PE에서 연산되는 부분 EV결과는 각 계층에 맞게 가산기에서 축적 가산되어 MB단위의 EV가 되고 비교기에서 이 값들을 비교하여 최소의 EV값을 갖는 MB를 찾게 된다. 이 때 화소 그룹의 크기가 증가(감소)할수록 가산 연산은 증가(감소)하고, 이후 비교 연산은 감소(증가)하게 되어 움직임 추정 전체 연산에 있어 축적 가산과 비교 연산의 합은 항상 일정하게 된다. 따라서 축적 가산과 비교 연산을 프로그램 가능하도록 하는 그림 7에 나타낸 바와 같은 비교기/가산기 유닛을 사용하였다.

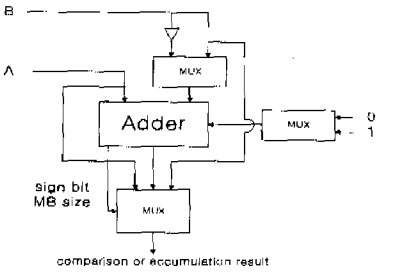


그림 7. 비교기/가산기 유닛

즉 가산기는 축적 가산 시에는 가산기로, 비교 연산 시에는 carry_in과 입력 B의 보수를 받아 뺄

셈기로 되어 최소 값을 출력시키도록 하여 경우에 따라 비교기와 가산기로 변형시켜 활용 가능하다. 그리고 축적가산 또는 비교 연산은 트리 구조를 사용하여 throughput을 향상시켰다. 최종적으로 MB 단위의 EV의 비교연산을 통하여 MEDV (Minimum Error Displacement Value)를 구하게 된다.

회로 합성은 64개의 PE를 갖는 PE 어레이를 기준으로, PE 어레이, 비교기, 가산기 트리에 대해 수행하였고, 그 결과를 표 4에 나타내었다. 회로 합성은 SYNOPSYS와 삼성 게이트 어레이 라이브러리를 사용하여 수행되었다.

표 4. 주요 모듈의 회로 합성 결과

	게이트 수
PE 어레이	66, 272
가산기 트리	5,840
비교기	2,436

V. 성능 분석 및 장점

제안된 아키텍처에서는 기존의 FBMA, HSA 알고리즘마다 서로 다른 연산 방법과 더불어 HSA의 다양한 계층에서 요구되는 서로 다른 데이터 패턴, 즉 MB 크기 N 및 연산 방법을 동일한 아키텍처로 효율적으로 연산할 수 있다. 이는 향후 객체 단위로 다양한 규격의 움직임 추정이 요구되는 MPEG 4에서도 적용 가능하다.

식 (3)에서 나타낸바와 같이 FBMA와 HSA 각 계층을 연산하는데 있어 동일한 연산량을 바탕으로 통일적인 아키텍처로 연산 가능하도록 설계되었으므로 FBMA와 HSA가 동일한 성능을 나타낸다. 이 경우에 HSA가 더 넓은 탐색 영역의 연산이 가능하다. 또한, 타 아키텍처의 성능과 비교해보면 하나의 CF MB에 대한 MEDV 계산시간이 $(N + 2p)^2$ 클록으로 N과 p에 의해 고정되는데 비해[3], 제안된 아키텍처는 성능정도에 따라 PE 수를 용이하게 조절할 수 있어서 여러 응용분야에 적용할 수 있다. 또한 모든 규격에 대하여 연속되는 MEDV 연산간의 휴지시간을 제거하여 PE의 이용도가 100%이므로 throughput을 극대화시킬 수 있다. 이는 특히 HDTV급과 같이 고화질의 경우 제안된 아키텍처가 더욱 우수한 성능을 나타낸다.

타 아키텍처와 비교하기 위하여, 성능 분석에 있어 HSA의 각 계층이 파이프라인으로 연결되어 있다고 가정한다. 설계된 PE에서는 PU를 u×v로 확장 할 수 있는데 여기서는 u = 2, v = 2를 사용하여 1 클록에 4×4 화소 그룹을 처리한다고 가정한다. 이 경우 화소마다 1/4 클록이 소요되고 총 계산되어야 할 화소 수는 전체 탐색 영역에 대하여 $(2p_{st} /S_{st})^2$이 된다. 예를 들어, p = 64인 경우, MEDV(i, j)의 연산 시간은 $(2\times64/1)^2/ 4 = 4,096$ 클록이 된다.

그림 8에 성능 분석 결과를 나타내었다. 그림 8의(c)에서 면적은 해당 아키텍처의 가산기와 비교기의 합을 나타내어 하드웨어 양과 성능과의 관계를 나타낸다. 본 성능 비교는 CCIR 규격(576 × 720 화소/프레임, 25 프레임/초)의 영상을 바탕으로 수행되었다. CF 연산간 지연시간에 있어서는 [5]와 [7]의 아키텍처와 같이 '0'을 나타낸다. CF 연산을 위한 클록 수에 있어서는 최소값을 나타내었고, 하드웨어 양과 연산 속도의 비교를 나타내는 Area와 연산 속도와의 관계에서는 [5], [7]과 같은 성능을 나타내었다. 그리고 45MHz의 클록을 사용한 경우 제안된 아키텍처가 가장 많은 프레임을 처리할 수 있다.

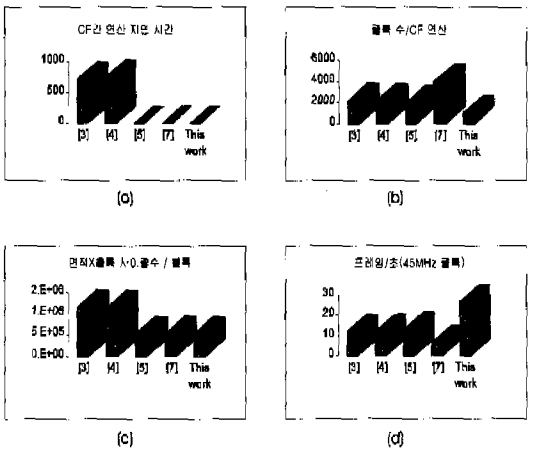


그림 8. 타 아키텍처와의 성능 비교
(a) CF간 연산 지연시간
(b) CF MB 연산 소요 클록 수
(c) 면적X클록 사이클 수/블록
(d) 프레임/초(45MHz 클록)

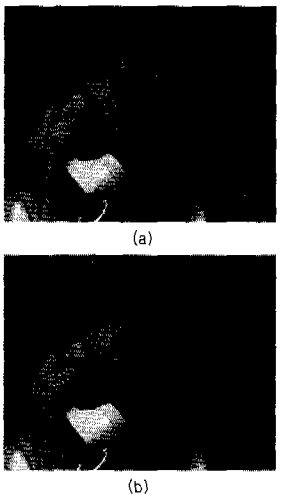
VI. 아키텍처 검증

실제 화상을 바탕으로 HDL로 검증할 경우 요구되는 막대한 시간을 감소시키기 위하여 C++을 사용하여 검증을 수행하였다. 또한 주요 모듈에 대해

VHDL을 사용하여 실제적인 검증을 수행하였다.

C++ 코드는 프로그램 가능 여부를 검증하기 위하여 제어신호에 해당되는 변수에 의해서만 동작이 제어되도록 하였다. 검증은 FBMA와 HSA에 대하여 수행하였는데, HSA1, 2, 3의 결과가 비슷하기 때문에 HSA로 표시하였다. 그림 9에 C++을 사용한 검증 결과를 나타내었다. C++검증 결과 FBMA의 PSNR은 29, MSE는 2.2e+2, HSA의 PSNR은 28, MSE는 2.3e+2을 나타내었다.

그림 10은 PE 어레이의 VHDL 검증 결과이다. sr_ext_in으로 표시되는 신호는 화상 데이터 버퍼를 통하여 PE 어레이로 입력되는 데이터이고, CF_in은 CF 데이터를 말한다. 그리고, ABS_top_sum와 ABS_bottom_sum은 필드/프레임 모드를 모두 지원하기 위해 필드 모드 단위로 절대값을 연산한 결과를 의미한다. 그리고 in_sel2_bit은 내부 SR의 제어 신호이다.



(a)

(b)

그림 9. C++ simulation 결과
(a) FBMA 결과 (b) HSA 결과

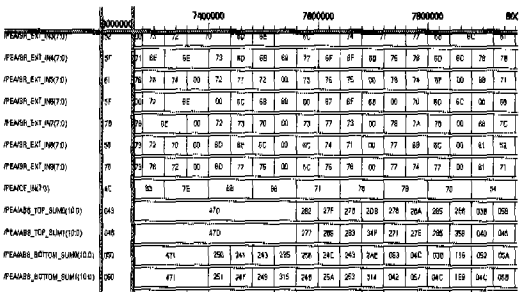


그림 10. PE 어레이의 VHDL 검증 결과

Ⅶ. 결 론

움직임 추정에 있어 대표적 알고리즘인 FBMA와 HSA를, 다양한 영상 규격에 있어 동일한 하드웨어로 연산 할 수 있고, MPEG 4에서 쓰이는 객체 단위 부호화에 활용 가능한 범용 움직임 추정 프로세서 아키텍처가 제안되었다. 각기 다른 화상 입력 데이터의 형태를 처리 할 수 있는 프로그램 가능한 I/O 인터페이스 유닛이 제안되었으며, 효율적인 내부 버퍼를 사용하여 I/O 핀 수의 최소화, 병목 현상 완화 및 MB 연산간의 지연시간으로 인한 throughput 저하 문제를 해결하였다. 또한 PE의 숫자와 연산 속도를 조절하여 다양한 영상 규격의 처리를 위한 성능 조절을 용이하게 하였다. 제안된 아키텍처를 검증하기 위하여 실제 영상을 바탕으로 모의 실험이 수행되었고 VHDL을 사용하여 주요 하드웨어 모듈의 동작을 검증하였다.

참 고 문 헌

[1] N. Hayashi, et. al., "Compact Motion estimator with a Simplified Vector Search Strategy Maintaing Encoded Picture Quality," *IEEE Custom Integrated Circuits Conference,* 1995.

[2] H. Yeo, Y. Hen, "A Novel Modular Systolic Array Architecture for Full-Search Block Matching Motion Estimation," *IEEE Trans. on Circuit and Systems for Video technology,* vol. 5, No. 5, Oct. 1995.

[3] C. H. Hsieh, T. P. Lin, "VLSI Architecture for Block-Matching Motion Estimation Algorithm," *IEEE Trans. on Circuit and Systems for Video technology,* vol. 2, No. 2, June 1992.

[4] K. Ishihara, S. Masuda, S. Hattori, H. Nishikawa, Y. Ajioka, T. Yamada, H. Amishiro, Yoshimoti, "A Half Precision MPEG-2 Motion-Estimation Processor with Concurrent Three-Vector Search," *ISCCC 95 Digest of Technical Papers,* pp. 288-289, 1995.

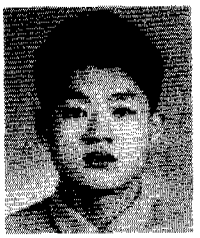
[5] Y. S. Jehng, L. G. Chen, and T. D. Chiueh, "An Efficient and Simple VLSI Tree Architecture for Motion Estimation Algori-

thm," *IEEE Trans. on Signal Processing,* vol. 41, No. 2, Feb. 1993.

[6] G. Gupta, C. Chakrabarti, "Architecture for Hierarchical and Other Block Matching Algorithms," *IEEE Trans. on Circuit and Systems for Video technology,* vol. 5, No. 6, Dec. 1995.

[7] K. M. Yang, M. T. Sun, L. Wu, "A Family of VLSI Design for the Motion compensation Block-Matching Algorithm," *IEEE Trans. on Circuit and Systems,* vol. 36, No. 10, Oct. 1989.

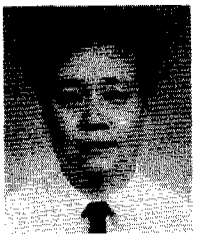
서 영 산(Young-san Seo) 준회원



1998년 2월 : 홍익대학교 전자공학과 졸업
1998년 3월~현재 : 홍익대학교 대학원 전자공학과 석사과정

<주관심 분야> 영상처리 VLSI 아키텍쳐 및 system 설계, IMT2000, shape adaptive DCT.

유 재 희(Jae-hee You) 정회원



1985년 2월 : 서울대학교 전자공학과 졸업
1990년 2월 : Cornell 대학교 전기공학과 박사
1990년 3월~1991년 3월 : Texas Instruments Dallas, VDL 연구원
1991년~1997년 : 홍익대학교 전자공학과 조교수
1997년~현재 : 홍익대학교 전자공학과 부교수

<주관심 분야> Image, Speech signal processing VLSI 알고리즘, 아키텍쳐 및 system 설계, Network IC, DRAM, SRAM 회로 설계