論文94-31B-9-9

완전탐색 블럭정합 알고리듬을 이용한 움직임 추정기의 VLSI 설계 및 구현

(Design and Implementation of Motion Estimation VLSI Processor using Block Matching Algorithm)

李 容 勳* **, 權 用 武*, 林 鎬 根*, 柳 根 壯*, 金 炯 坤*, 李 文 基**

(Yong Hoon Lee, Yong-Moo Kwon, Ho Geun Lim, Keun Jang Ryoo, Hyoung Gon Kim and Moon Key Lee)

要 約

본 논문은 움직이는 영상을 추정하여 변위벡터를 찾아내는 움직임 추정기에 관한 것으로 완전탐색 블럭정합 알고리듬을 이용한 움직임 추정기의 구조와 VLSI 설계에 대한 것이다. 본 논문에서 제안한 구조의 특징은 양방향 병렬 파이프라인 처리구조로서 블럭정합 처리시 PE(Processing Element) idle time을 감소시켰다. 이로인해 동일한 클럭 주파수에서 기존 구조에 비해 처리 속도를 개선하였다. 또한 HSPICE 모의시험을 근거로하여 PE가 75 MHz 클럭 주파수에서 동작되도록 설계하였다.

Abstract

This paper presents a new high-performance VLSI architecture and VLSI implementation for full-search block matching algorithm. The proposed VLSI architecture has the feature of two directional parallel and pipeline processing, thereby reducing the PE idle time at which the direction of block matching operation within the search area is changed. Therfore, the proposed architecture is faster than the existing architectures under the same clock frequency. Based on HSPICE circuit simulation, it is verified that the implemented processing element is operated successfully within 13 ns for 75 MHz operation

Ⅰ. 서 론

오늘날 영상압축 방식에서 많이 연구되고 있는 움직임 추정 방법은 크게 나누어 블럭정합 알고리듬 방식과 화소 반복 알고리듬으로 분류할 수 있다.[1] 블럭정합 알고리듬 방식은 영상을 여러개의 작은 블럭으로 나누어 각 블럭에 대한 변위벡터를 구하여 전송하는 방법이며[2-5] 화소 반복 알고리듬 방식은 각 화소에 대하여 회귀적 방법을 이용하여 움직임 벡터를 구하는 방법이다.[6,7] 이 두 방법 모두 변위벡터를 구하는데 있어서 많은 계산량이 소요하므로 하드웨어 구현이 필수적이라 할 수 있다. 그러나 하드웨어 구현에 있어서 블럭정합 알고리듬이 화소 반복 알고리듬보다 간단하고 계산량이 작아서 하드웨어 설계가 쉬우며 또한 변위벡터를 구하는 속도가 빨라 현재 많이 사용되고 있다.

*正會員, 韓國科學技術研究院 情報電子研究部 (KIST)

**正會員, 延世大學校 電子工學科 (Dept. of Elec. Eng., Yonsei Univ.)

接受日字 : 1993年 12月 15日

완전 탐색 블럭 정합 알고리듬의 실시간 구현을 위한 VLSI 구조로서 참고문헌 [2] 에서는 시스토릭 어레이에 근거한 구조를 제시하고 각 구조의 성능을 고찰하였다. 참고문헌 [2] 에서 제시한 구조는 블럭 정합시 이용하는 병렬성 측면에서 블럭 화소 병렬 처리 구조와 변위벡터 병렬 처리 구조로 분류할 수 있다.[1] 그러나 참고문헌 [2] 에는 프레임 메모리에 존재하는 탐색영역 데이타를 중복 접근하지 않고 효과적으로 PE(Processing Element) 어레이에 전달하는 탐색영역 메모리 구조에 대한 고려가 되어 있지 않다.

현재 VLSI 칩으로 구현되어 이용되고 있는 움직임 추정기의 경우 참고문헌 [2] 에서 제시한 구조와 같이 블럭 화소 병렬 처리 구조[8] 와 변위벡터 병렬처리 구조[9] 로 분류할 수 있다. 이들 구조는 프레임 메모리로 부터 탐색영역 데이타를 PE 어레이에 전달 時 한 방향 병렬 파이프라인 구조를 가짐으로인해 많은 PE idle time 이 존재하는 단점을 가진다.

본 논문에서는 데이타 전달과정에서 생기는 PE idle time을 감소시키기 위해 양방향 병렬 파이프라인 구조에 근거한 움직임 추정기 구조를 설계하였다. 설계된 움직임 추정기 구조는 변위벡터 병렬 처리 구조에 근거하며 PE idle time의 감소로 인해 같은 클럭 주파수에서 기존의 움직임 추정기[8-9] 보다 약 1.7 배가 빠르다.

Ⅱ장에서는 기존의 움직임 추정기 구조를 고찰하고 Ⅲ장에서는 양방향 병렬 파이프라인 구조에 근거한 움직임 추정기 구조를 제시하고 기존 구조와 성능을 비교한다. Ⅳ장에서는 움직임 추정기의 layout 설계에 대해 기술한다. Ⅴ장에서는 설계된 PE 회로의 모의시험 결과에 대해 고찰한다.

Ⅱ. 기존의 움직임 추정기 구조

본 논문에서는 기준블럭의 크기가 nxn (여기서 n=8) 이며 움직임 추정기가 8x8의 PE 어레이로 구성되어 지는 경우에 대해 고찰한다. 또한 탐색영역의 크기는 움직임벡터의 속도를 -8/+7 로 하여 23x23 개의 화소로 구성되어 있다.

1. 기존의 움직임 추정기 구조 Ⅰ [8]

그림 1에 나타낸 구조의 동작 순서는 먼저 PE 어레이 내에 기준블럭의 화소값을 n^2 사이클 동안 저장해 놓고 탐색영역에서는 라인 단위로 (하나의 라인은 8 개의 화소로 구성된다.) 순차적으로 PE 어레이에 입력된다. 여기서 주목할 점은 하나의 라인에 존재하는 각각의 화소 데이타는 8x8 PE 어레이에서 해당되는 열에 존재하는 8개 PE에 공통적으로 입력된다. 이때 PE 어레이에서 각각의 PE는 PE에 저장되어 있는 기준블럭의 화소값과 입력되는 탐색영역의 화소값과의 절대차를 계산하고 이 절대차를 지금까지 PE 내부에 누적된 절대차 누적 합과 더한 후 그 다음 라인의 PE로 보낸다. PE 어레이 마지막 라인에서는 계산된 값들을 부분합 처리기에 보내며 여기서 마지막 라인에 있는 8개 PE의 부분 누적 합 (partial accumulated sum) 들이 모두 더해지게 되어 하나의 탐색블럭에 대한 변위값이 계산된다.

이 구조는 기준블럭의 데이타가 순차적으로 각 PE에 완전히 입력되기 위하여 기준블럭 크기의 클럭 사이클이 요구되며, 또한 탐색영역내에서 비교 대상 블럭의 위치가 수평 이동할 때 마다 n-1 클럭 사이클의 PE idle time 이 존재하며 그 이후 부터는 매 클럭 사이클마다 하나의 탐색블럭에 대한 변위값이 계산된다. 따라서 탐색 영역 안에 있는 전체 256 블럭에 대한 변위값을 구하기 위해서는 432 클럭 사이클 [64+(7+16)x16] 이 요구된다. 이 구조의 단점은 초기 기준블럭의 데이타를 PE 어레이에 저장하기 위하여 n^2 사이클의 PE idle time 과 탐색영역내에서 비교 대상 블럭 위치가 수평 이동시마다 n-1 사이클의 PE idle time 이 존재한다는 점이다.

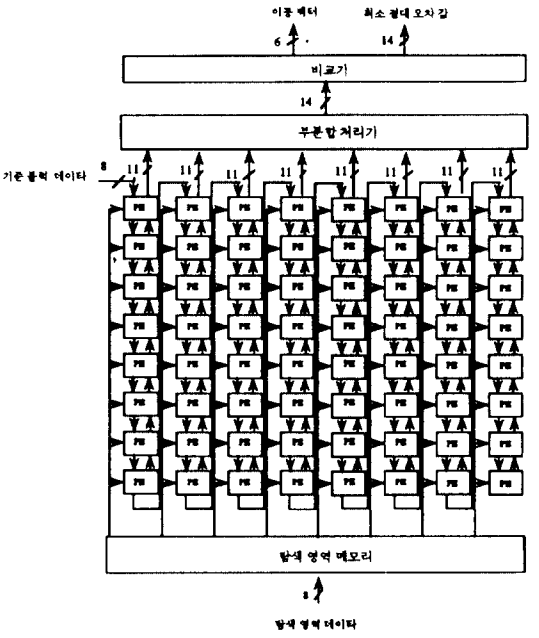


그림 1. 기존의 움직임 추정기 구조 Ⅰ
Fig. 1. Architecture Ⅰ of motion estimator.

2. 기존의 움직임 추정기 구조 Ⅱ [9]

그림 2는 그림 1과는 달리 기준 블럭의 화소값을

미리 PE 어레이 내에 저장하지 않고 순차적으로 모든 PE에 동시에 전달되는 구조로서 한 클럭 사이클에 탐색영역의 데이타가 한 라인씩 순차적으로 PE 어레이에 입력되고 이동되며 이때 각 PE에서는 기준블럭의 데이타와 각 탐색블럭의 데이타에 대한 절대차를 구하여 누적하고 누적값을 자신의 PE 에 기억한다.

이 구조는 120 (15x8=120) 클럭 사이클에 64개의 블럭에 대한 변위값이 동시에 계산되어 출력되며 (즉 각각의 PE는 하나의 탐색블럭에 대한 하나의 변위값 (절대차의 누적값)을 구한다) 탐색영역에 속하는 256개의 탐색블럭에 대한 변위값들을 구하기 위해서는 4번의 반복이 요구되어 480 클럭 사이클 (15x8x4) 이 요구된다.

이 구조는 PE의 설계가 용이하며, 제어도 간단하여 전체 칩의 크기를 줄일 수 있다는 장점이 있지만 많은 클럭 사이클이 요구되는 단점을 갖는다.

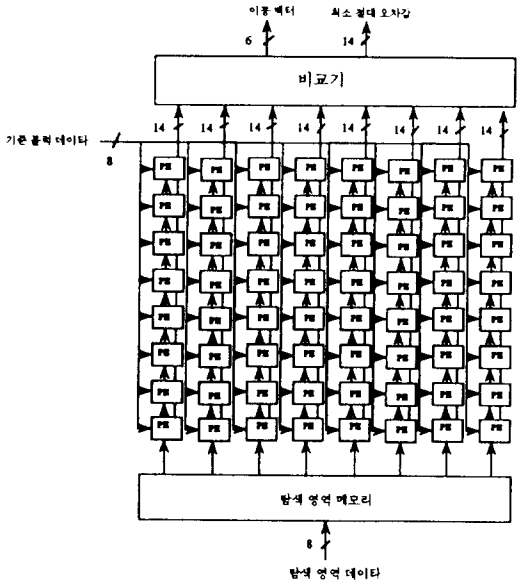


그림 2. 기존의 움직임 추정기 구조 Ⅱ
Fig. 2. Architecture Ⅱ of motion estimator.

Ⅲ. 새로운 움직임 추정기 구조

본 논문에서 제안하는 움직임 추정기의 구조는 그림 3에 나타나있으며, 기존의 움직임 추정기 구조 II를 개선한 구조이다. 기준블럭의 크기는 8x8이며 탐색영역의 크기는 23x23으로, 움직임 벡터의 범위는 수직 수평 양방향이 같이 -8/+7로 하여 한번에 64개의 PE가 64개의 탐색블럭에 대한 변위값을 추출하게 되며, 이러한 과정을 네번 수행함으로써 전체 탐색영역에 대하여 256개의 변위값을 구하게 된다. 본 논문에서는 용어의 혼란을 피하기 위하여 PE 어레이의 64개 PE가 64개의 탐색블럭에 대한 누적값을 추출하는 하나의 과정을 처리과정이라 한다. 따라서 본 논문에서 제안한 움직임 추정기는 전체 탐색영역에 대한 256개의 탐색블럭의 누적값을 구하기 위해서는 네번의 처리과정을 거쳐야 한다.

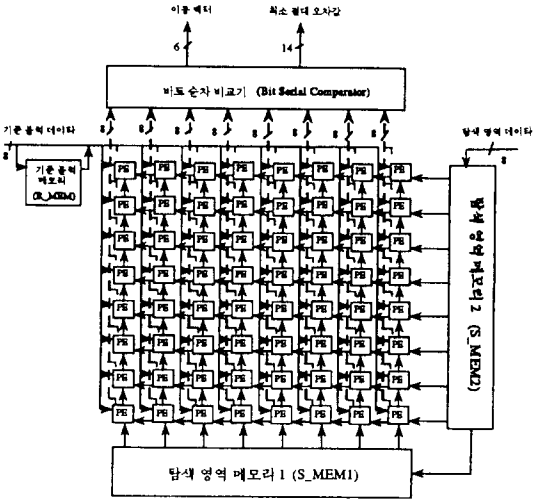


그림 3. 제안하는 움직임 추정기 구조
Fig. 3. Architecture of proposed motion estimator.

그림 4는 현재 프레임의 기준블럭과 이전 프레임에 존재하는 탐색영역을 나타낸다. 실제 탐색영역은 23x23이나 데이타 전송 클럭 사이클 시간의 규칙성을 고려하여 24x24 크기의 메모리를 탐색영역으로 설정한다. 또한 설명의 편의를 위해 그림 4에서 24x24 크기의 탐색영역을 8x24 크기의 3개의 부영역으로 구분하였다.

본 논문의 움직임 추정기의 동작을 간단히 설명하면, PE 어레이의 각 PE들은 두개의 클럭제어신호 (V0, V1)에 의하여 탐색영역 메모리1(S_MEM1) 또는 탐색영역 메모리2(S_MEM2)에 있는 데이타가 이차원 PE 어레이의 수직 또는 수평방향으로 입력되고, 첫번째와 세번째 처리과정의 클럭 V0의 초기 7 사이클과 두번째와 네번째 처리과정의 마지막 1 사이클만 제외하고 매 사이클마다 기준블럭의 데이타는 모든 PE에 동시에 가해져서 두 입력 데이타의 차의 절대값을 누적하게 되며, 한번의 처리과정이 끝나면

64개의 탐색블럭에 대한 누적된 64개의 변위값이 각 PE의 누적기(CSA2:Carry Save Accumulator)로부터 16 비트 레지스터로 이동되고 각 PE의 16 비트 레지스터 값은 한 비트씩 상위 비트부터 순차적으로 클럭 V0의 사이클에 의해 비교기로 출력된다. 따라서 본 논문에서 제안한 움직임 추정기의 구조는 첫번째와 세번째 처리과정에서의 클럭 신호 V0의 초기 7 사이클과 두번째와 네번째 처리과정의 마지막 1 사이클만 제외하고는 전혀 idle time이 없으므로 클럭신호 V0의 272 [2x(7+1+2x64)=272] 사이클만에 256개의 탐색블럭에 대한 변위값을 구할 수 있는 고속 움직임 추정기 구조이다.

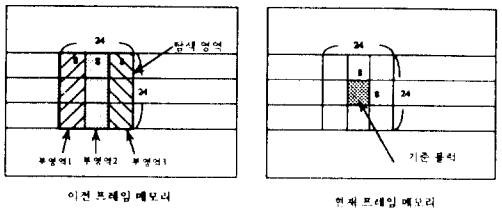


그림 4. 기준블럭, 탐색영역 및 부영역의 개념
Fig. 4. Concept of reference block, search area and sub area.

1. PE의 구조

본 논문에서 제안하는 움직임 추정기의 PE는 그림 5에 나타낸 바와 같이 PE 어레이가 수직 및 수평방향의 병렬 파이프라인 구조를 취함으로 양 방향의 데이타를 지연시키기 위한 두개의 지연소자(기존의 움직임 추정기 구조 Ⅱ에서는 한 방향 병렬 파이프라인 구조를 취함으로 한개의 지연소자만 있었음) 와 양 방향의 데이타 값중에 하나를 선택하기 위한 멀티플렉서를 새롭게 포함시킨 변형된 구조의 PE이다. 본 논문의 PE에 대한 동작을 설명하면 다음과 같다.

(1) 기준블럭 데이타의 PE 입력 방법 :
클럭 V0가 positive edge일때 첫번째와 세번째 처리과정의 초기 7 사이클 동안 과 두번째와 네번째 처리과정의 마지막 1 사이클만 제외하고 기준블럭의 데이타 가 모든 PE의 CSA1 (Conditional Sum Adder)으로 동시에 입력된다. 이때 기준블 럭의 데이타는 첫번째 처리과정에서는 외부로부터 입력되지만 두번째와 세번째 그리고 네번째 처리과정은 움직임 추정기내에 있는 기준블럭 메모리로부터 입력 된다.

(2) 탐색영역 메모리 데이타의 PE 입력 방법 :
클럭 V1이 high 일때 : 클럭 V0가 positive edge 이면 S_MEM1에 있는 데이타가 그림 5에 나타나 있는 PE의 레지스터1으로 입력되고, 또한 멀티플렉서를 통 하여 레지스터2 및 CSA1 에 입력된다. 기존의 레지스터1에 있던 데이타는 다음 단(수직 방향) PE의 레지스터1 및 멀티플렉서를 통하여 레지스터2와 CSA1으로 입력된다.
클럭 V1이 low 일때 : 클럭 V0가 positive edge 이면 S_MEM2에 있는 데이타가 멀티플렉서를 통하여 레지스터2와 CSA1에 입력되고, 기존의 레지스터2에 있던 데이타는 다음단(수평 방향) PE의 멀티플렉서를 통하여 레지스터2와 CSA1으로 이동하게 된다. 멀티플렉서는 두개의 입력신호중 하나를 클럭 V1에 의해 선택 하여 (본 움직임 추정기에서는 클럭 V1이 high이면 S_MEM1 또는 이전 단의 레지스터1에 있는 데이타를, 클럭 V1이 low이면 S_MEM2 또는 이전 단의 레지스터2에 있는 데이타를 선택한다) CSA1과 레지스터2로 보내주는 역할을 한다.

각 PE는 CSA1에 의하여 기준블럭의 데이타와 탐색블럭의 데이타에 대한 차를 구하고 XOR (Exclusive OR)를 통하여 절대값을 구한 후에, CSA2(Carry Save Accumulator)에 계속 누적하게 된다. 64개의 변위벡터를 구하면 각 PE의 CSA2에 누적된 값들은 14 비트 레지스터로 이동하고 이값은 1 비트씩 비교기로 출력하게 되며 CSA2의 누적값은 0 으로 clear된다.

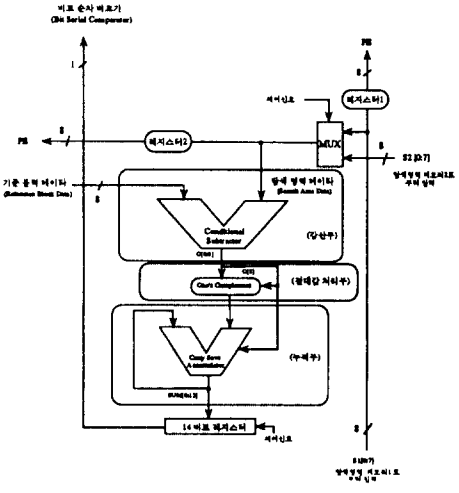


그림 5. 본 논문에서 제안하는 PE의 구조
Fig. 5. Architecture of proposed processing element.

2. 메모리부의 구조

본 논문에서 제안하는 메모리의 구조는 그림 6에 나타낸바와 같이 기준블럭의 데이타를 저장하기 위하여 64개의 8비트 레지스터로 구성된 기준블럭 메모리(R_MEM)와 탐색영역의 부영역 1 데이타를 저장하기 위한 192개(8x24)의 8비트 레지스터로 구성된 S_MEM1과 부영역 2의 데이타를 저장하기 위한 192개(8x24)의 8비트 레지스터로 구성된 S_MEM2 및 8개의 8 비트 레지스터로 구성된 하나의 버퍼로 구성되어 진다.

그림 7은 프레임 메모리내의 탐색영역, 움직임 추정기 내부의 탐색영역 메모리 및 PE 어레이간의 화소 데이타 전달 과정을 나타낸 것이다. 그림 7에 나타낸 바와 같이 탐색영역 데이타를 3개의 부영역으로 나누고 각각의 부영역을 다시 3개의 8x8 데이타로 나누어 설명한다. (1) 초기상태에 S_MEM1과 S_MEM2에 저장되어 있는 탐색영역 데이타를 나타낸다. 이때 PE 어레이내에는 탐색 대상 데이타가 존재하지 않는다. (2) 첫번째 처리과정의 클럭 V0 의 8 사이클이 지난 후의 상태이다. PE 어레이 내부에는 8x8 데이타 1A가 입력되었다. (3) 첫번째 처리과정이 끝난 후의 상태로서 S_MEM2에는 프레임 메모리에 존재하는 탐색영역 3A가 입력되었다. (4) 두번째 처리과정이 끝난 후의 상태로서 S_MEM2에 3B가 입력되었다. (5) 세번째 처리과정에서 초기 V0 클럭 8 사이클로서 이 기간 중에는 S_MEM2에 있는 데이타는 수직 방향으로 쉬프트되나 외부 프레임 메모리로 부터는 데이타가 입력되지 않는다. (6) 세번째 처리과정이 끝난 후의 상태로서 3C가 S_MEM2에 입력되었다. (7) 네번째 처리과정이 끝난 후의 상태로서 S_MEM1 및 S_MEM2내에는 초기상태와 같은 형태로 부영역 2 및 부영역3가 입력되었음을 알 수 있다. 따라서 다음번 탐색영역의 처리시 프레임 메모리로 부터 데이타를 중복 접근하지 않고 연속적으로 움직임 추정 처리를 수행할 수 있다.

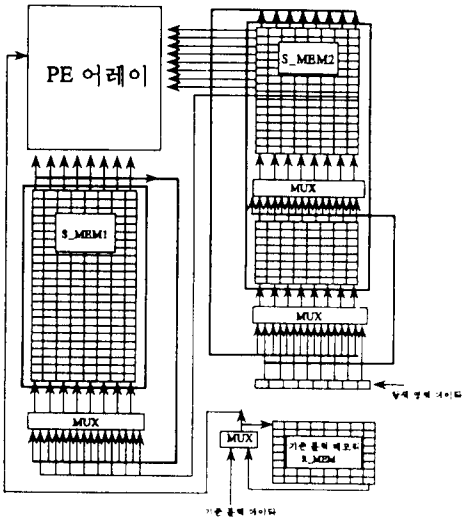


그림 6. 메모리 블럭의 구조
Fig. 6. Architecture of memory block.

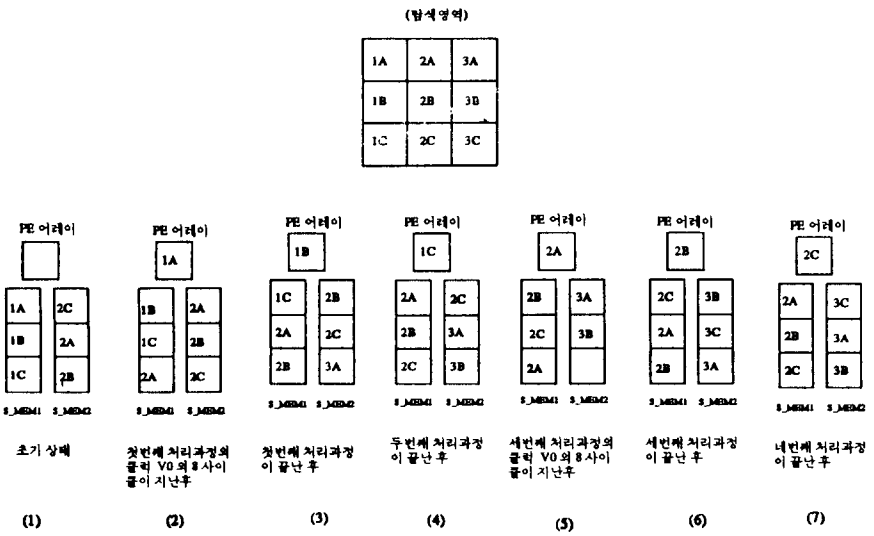


그림 7. 탐색영역 메모리의 동작과정
Fig. 7. Data flow of search area memory.

3. 비교기부의 구조

본 논문에서 제안하는 움직임 추정기의 구조는 변위벡터 병렬처리형이므로 각 PE가 하나의 탐색블럭에 대한 변위값을 가지게 된다. 따라서 하나의 처리과정이 끝날 때 동시에 64개의 변위값이 출력되며 그 중에 가장 작은 변위값을 다음 처리과정이 끝나기 전에 추출해야한다. 본 논문에서는 요구되는 하드웨어 및 처리의 효율성을 고려하여 비트 순차형 비교기 구조를 설계하였다. 설계한 구조는 8개의 PE에 비트 순차형 비교기(본 논문에서는 수직 비교기라 함)를 하나씩 설치하여 64개의 PE에 8개의 수직비교기로 8개의 가장 작은 변위값과 수직 변위벡터를 구하도록 하였으며, 또 하나의 다른 비트 순차형 비교기(본 논문에서는 수평비교기라 함)로 두어 8개의 수직비교기에서 구한 8개의 가장 작은 변위값들과 전 처리과정에서 구한 가장 작은 변위값중에서 제일 작은 변위값 및 변위벡터를 구하도록 하였다.그림 8은 8개의 수직비교기와 하나의 수평 비교기로 구성된 비교기부 구조이다.

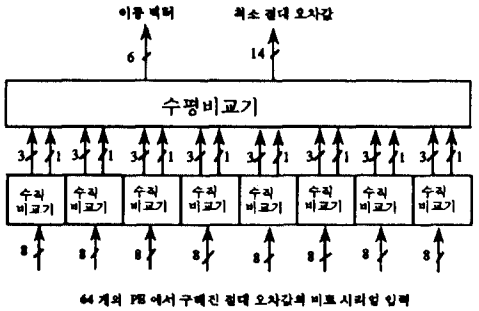


그림 8. 비교기부의 구성
Fig. 8. Configuration of comparator module.

4. 제어부

제어부는 탐색 메모리로 부터 PE 어레이로 데이타 이동을 제어한다. 그림 9는 클럭 V0 및 제어신호 V1에 따라 탐색영역 메모리로부터 PE 어레이로 데이타가 이동하는 과정을 나타낸다. V0 클럭의 positive edge에서 V1 제어신호가 high이면 S_MEM1로부터 PE 어레이로 데이타가 이동하며 V1 제어 신호가 low이면 S_MEM2로부터 PE 어레이로 데이타가 이동한다.

한편 블럭 정합시 기준 블럭의 위치 형태는 그림 10에 표시한 바와 같이 9가지 경우로 분류할 수 있다. 5번 위치인 경우는 탐색영역 전체가 이전 프레임 메모리에 존재하나 그 이외의 경우는 탐색영역이 일부만 존재하므로 존재하지 않는 부분은 화소 값을 0으로 하여 움직임 추정을 수행하는 기능이 요구된다.

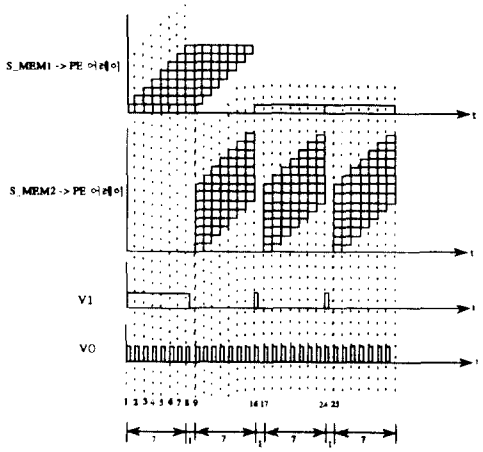


그림 9. 데이타 이동 제어
Fig. 9. Control of data transfer.

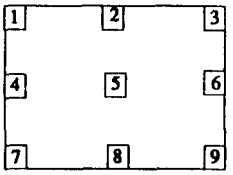


그림 10. 기준블럭 위치 분류
Fig. 10. Classification of reference block position.

5. 처리성능 평가 및 비교

본 논문에서 제안한 움직임 추정기의 구조와 기존의 움직임 추정기와 비교를 표 1에 나타내었다. 본 논문에서는 PE 어레이가 양방향 병렬 파이프라인 처리를 실행함으로써 기존의 움직임 추정기보다 동일한 클럭신호에 대하여 약 1.7배 가량 빠른 새로운 구조의 움직임 추정기 구조와 이에 근거한 처리기를 설계하였다.

표 1. 움직임 추정기의 비교
Table 1. Comparison of motion estimator.

	구조 I[9]	구조 II[10]	새로운 구조
기준블럭의 크기	8 × 8	8 × 8	8 × 8
탐색영역의 크기	23 × 23	23 × 23	23 × 23
움직임 속도(p)	8	8	8
PE의 수	8 × 8	8 × 8	8 × 8
클럭 사이클 수	432	480	272
H/W Overhead (PE 당 Latch 수)	1	1	2
메모리 크기(bytes)	24 × 24	24 × 24	24 × 16

Ⅳ. Layout 설계

본 논문에서 제안한 움직임 추정기의 VLSI 구조를 바탕으로 완전 탐색 블럭정합 알고리듬의 고속 처리가 가능토록 하기 위한 움직임 추정기의 회로 및 Layout 설계에 대해 기술한다.

1. Conditional SUM Adder 설계

본 PE는 클럭 주파수가 75 MHz 에서 실시간 처리가 가능하도록 하기 위해 속도가 빠른 Conditional Sum Adder를 사용하여 설계하였다.

2. Dynamic Carry Save Accumulator와 CMOS 래치 설계

본 PE의 누적기 설계에 있어서는 ripple carry 가산기의 carry 신호의 전파지연을 줄이기 위하여 2 Phase Clock 동적 회로를 사용한 3 입력 전가산기와 두개의 정적 래치로 구성된 Dynamic Carry Save Accumulator를 이용하였다.

3. PE의 Layout 설계

PE는 앞서 설명한 바와 같이 Conditional Sum Adder(CSA1), 절대값을 구하기 위한 Excluve OR, Carry Save Accumulator(CSA2)및 지연소자로 구성된다. 그림 11은 PE가 8x8 로 구성된 PE 어레이의 layout 을 나타내며 표 2 는 설계된 움직임 추정기의 주요 특징이다.

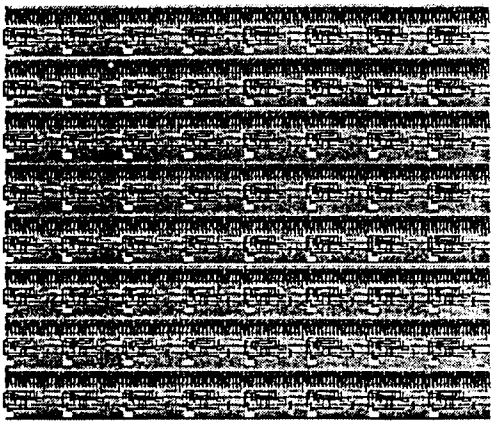


그림 11. 설계된 움직임 추정 PE 어레이의 레이아웃도

Fig. 11. Layout of designed motion estimation PE array.

표 2. 설계된 움직임 추정기의 주요 특징

Table 2. Main features of designed motion estimation processor.

Technology	1.2 um double-metal CMOS
Number of Transistor	145 K
Max. Clock Frequency of PE	75 MHz
PE 어레이 (8x8) 크기	8.0 x 6.9 mm^2

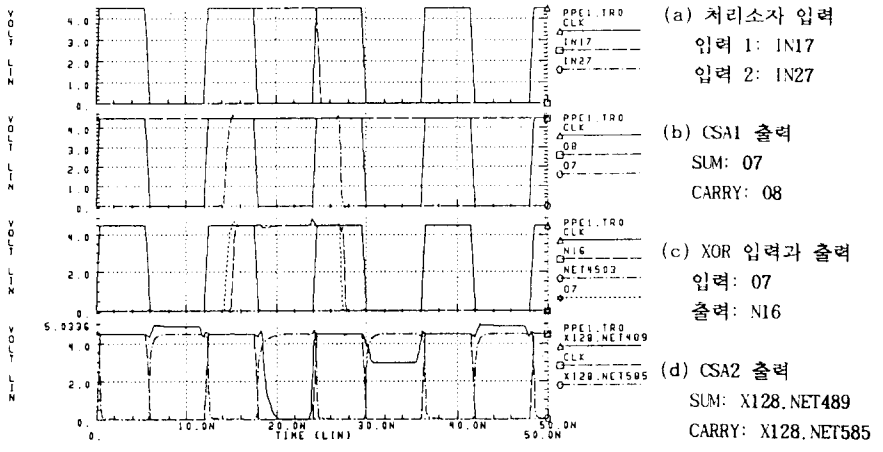


그림 12. HSPICE 회로 모의시험

Fig. 12. HSPICE circuit simulation

Ⅴ. 모의시험 및 검토

본 논문에서 설계한 PE를 75MHz 클럭 속도로 처리가 가능하기 위해서는 기본 클럭의 주기가 13.3 ns 보다 작아야 한다. 본 논문에서는 설계의 안정성을 고려하여 1 클럭 사이클 시간을 12 ns 로 하였다. 그림 12는 HSPICE 를 사용하여 단위 처리기 소자에 대한 회로 모의시험을 수행한 결과이다. 그림 12(a)는 conditional sum adder (CSA1)에 입력이 가해지는 것을 나타낸다. 그림 12(b)는 CSA1의 출력지연시간(2 ns)을 나타낸다. 그림 12(c)는 절대값을 구하는 회로의 지연시간(1 ns)을 나타낸다. 그림 12(d)는 dynamic 회로로 구성된 carry save accumulator(CSA2)의 지연시간(2 ns)를 나타낸다. 전체적으로 입력이 가해지고 약 8 ns 후에는 결과값을 구할수 있다.

그림 12에 나타낸 바와 같이 본 논문에서 설계된 PE 는 8 ns 내에 동작됨을 알 수 있다. 한편 본 논문에서 제안된 구조는 8x8 PE 어레이에 기준 블럭 데이타가 동시에 인가되는 구조를 가지므로 다음 사항을 고려해야 한다. 1) 각 경로의 길이가 상이함으로 인해 경로 지연 시간이 다르게 된다. 이에대한 처리는 각각의 처리기 소자에서 연산 결과를 래치 시키는 파이프라인 처리에 의해 동작을 안정화 시킬 수 있다. 2) 경로 길이로 인한 지연시간을 고려하여 가능한 경로 지연시간을 감소시키는 것이 요구된다. 즉 요구되는 칩의 동작 성능을 고려하여 처리기 소자의 동작 성능과 경로 지연시간을 함께 고려하여야 한다. 경로 지연 시간을 감소시키는 방법으로서는 기준 블럭 데이타 출력시 다단 출력 버퍼를 사용하고 또한 sheet resistance 와 capacitance가 적은 metal2 를 연결 path 로 사용하는 것을 고려할 수 있다.

Ⅳ. 결 론

본 논문에서는 동 영상에 대한 움직임 추정 알고리듬의 하나인 완전탐색 블럭정합 알고리듬을 이용한 고속 움직임 추정기의 효율적인 VLSI 구조를 제안하고 이에 사용되는 PE를 설계하였다. 본 논문에서 제안한 구조는 기존의 움직임 추정기보다 동일한 클럭 주파수를 사용할 경우에 모의시험 결과 약 1.7배 이상 빠르며, 또한 고속 움직임 추정기의 설계를 목표로 함으로써 PE에 있어서 Conditional Sum Adder를 사용하여 13 [ns] 이하에서 두개의 8 비트 데이타의 감산, 절대값, 누적 연산이 가능한 PE를 설계하였으며 이의 HSPICE 회로 모의시험을 통하여 설계의 정확성을 확인하였다.

본 논문에서 제안한 구조는 8x8 의 블럭 크기에 -8/+7 의 움직임 범위를 지원하는 구조이다. 그러나 일반적으로 16x16 의 블럭 크기에 더 큰 움직임 범위의 지원이 요구된다. 이의 처리를 위해서는 본 논문에서 제안된 구조의 확장 연구가 요구되며 여러개의 칩을 사용하여 병렬처리하는 구조에 대한 고려가 요구된다.

參 考 文 獻

[1] "비디오 코덱용 영상압축 VLSI 처리기 개발(Ⅱ)에 관한 연구", 연구보고서, 한국과학기술연구원, 1993

[2] T. Komarek, P. Pirsch, "Array Archtectures for Block Matching Algori- thms," IEEE Trans. Circuits & Systems, Vol 36, No. 10, pp. 1301-1308, Oct. 1989.

[3] K.M. Yang, M.T. Sun, L. Wu, "A Family of VLSI Designs for the Motion Compensation Block-Matching Algorithm," IEEE Trans. Circuits & Systems, Vol. 36, No. 10, pp. 1317-1325, Oct. 1989.

[4] L.D. Vos, M. Stegherr, "Parameterizable VLSI Architectures for Full-Search Block-Matching Algorithm," IEEE Trans. Circuits & Systems, Vol 36, No. 10, pp. 1309-1316, Oct. 1989.

[5] "동영상 압축부호화 기술(III)," 한국과학기술원, 1993

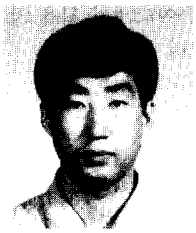
[6] E.D. Fimout, J.N. Driessen, E.F. Deprettere, "Parallel Architecture for a Pel-Recursive Motion Estimation Algorithm," IEEE Trans. Circuits & Systems for Video Tech., Vol. 2, No. 2, pp. 159-168, June 1992.

[7] R.C. Kim, S.U. Lee, "A VLSI architecture for a Pel-Recursive Motion Estimation Algorithm," IEEE Trans. Circuits & Systems, Vol. 36, No. 10, pp. 1291-1300, Oct. 1989.

[8] STV3220 Motion Estimation Processor, SGS-THOMSON, May. 1989.

[9] L64720 Video Motion Estimation Processor, LSI LOGIC, Sep. 1991.

著 者 紹 介



李 容 勳(正會員)
1957年 3月 20日生. 1985年 연세대학교 전기공학과(학사). 1994年 연세대학원 전자공학과(석사). 1984年 ~ 현재 금성일렉트론 선임연구원. 주관심 분야는 반도체 및 시스템 설계, DSP 및 영상시스템, 디바이스 모델링 등임.

•

林 鎬 根(正會員) 第 31卷 A編 第 6號 參照

權 用 武(正會員) 第 30卷 B編 第 10號 參照
현재 한국과학기술연구원 선임연구원

•

柳 根 壯(正會員) 第 31卷 A編 第 6號 參照

•

金 炳 坤(正會員) 第 26卷 第 11號 參照
현재 한국과학기술연구원 책임연구원

•

李 文 基(正會員) 전자공학회지 第 20卷 第 11號 參照
현재 연세대학교 전자공학과 교수