論文94-31B-7-15

가산투영을 이용한 2단계 고속 블럭정합 알고리듬의 하드웨어 설계

(Hardware Design of a Two-Stage Fast Block Matching Algorithm Using Integral Projections)

潘 聲 範*, 蔡 承 秀*, 金 俊 植**, 朴 來 弘**, 趙 威 德***, 林 信 一***

(Sung Bum Pan, Seung Soo Chae, Joon-Seek Kim, Rae-Hong Park, We-Duke Cho, and Shin-Il Lim)

要 約

본 논문에서는 동영상의 블럭정합 알고리듬의 하드웨어 구현에 대하여 고찰하였다. 움직임 추정 기법중 전역탐색 블럭정합 알고리듬에 비하여 성능이 비슷하고 계산량을 현저하게 줄인 가산투영을 이용한 2단계 블럭정합 알고리듬의 시스톨릭 어레이를 이용한 하드웨어 구조를 제안하였다. 제안한 구조는 전역탐색 블럭정합 알고리듬의 하드웨어 구조보다 2~15배이상 빠르게 동작한다. 전역탐색 블럭정합 알고리듬과 가산투영을 이용한 2단계 고속 블럭정합 알고리듬의 구현을 위해 각각의 구조들을 SPW와 VHDL을 이용하여 모델링하고 시뮬레이션한 결과를 보였다.

Abstract

In this paper, we investigate the hardware implementation of block matching algorithms (BMAs) for moving sequences. Using systolic arrays, we propose a hardware architecture of a two-stage BMA using integral projections which reduces greatly computational complexity with its performance comparable to that of the full search (FS). Proposed hardware architecture is faster than hardware architecture of the FS by 2 ~ 15 times. For realization of the FS and two stage BMA, modeling and simulation results using SPW and VHDL are also shown.

Ⅰ. 서 론

통신기술의 발달은 음성정보의 전달에서 영상정보의 전달로 진화되어 왔으며, 앞으로의 정보화 시대에서 영상정보를 이용한 정보통신 사업의 발달은 통신기술의 발전 속도와 그 맥을 같이 할 것이다. 이에, 제한된 대역폭과 메모리의 효율을 극대화시킬 수 있도록 영상 데이타를 압축하는 기술이 고품질의 영상정보 서비스를 위해 필수적이다.

동영상 전송 시스템의 중요개발부분인 영상신호처리부에는 영상취득, 움직임 검출 및 보상, 예측오차 부호화, 가변길이 부호화 등이 있는데, 가장 중심이 되는 것이 움직임 검출 및 보상이다. 이의 성능에 따라 전체 시스템의 성능이 좌우되므로 이에 대한 연구

* 準會員, ** 正會員, 西江大學校 電子工學科
(Dept. of Electronic Eng., Sogang Univ.)
*** 正會員, 電子部品綜合技術研究所
(Korea Electronics Technology Institute)
※ 본 연구는 전자부품종합기술연구소의 연구비 지원으로 이루어진 것임
接受日字 : 1993年 11月 16日

가 중요하며, 이 부분에서의 계산량도 다른 부분에 비해 방대하므로 효율적인 동영상의 실시간 전송을 위해 움직임 검출 및 보상부분에 대한 VLSI 작업이 필수적이다.

동영상 부호화에 있어 많이 사용되는 움직임 보상 부호화 (Motion Compensated Coding)[1-2]는 움직임을 검출하여 이에 따른 시간적인 중복성을 제거하고, 움직임을 나타내는 이동벡터와 움직임 보상에 따른 예측오차를 전송하여 영상데이타를 압축하는 기법이다. 움직임을 검출하는 기법중 블럭정합 알고리듬 (BMA : Block Matching Algorithm)[1]은 다른 방법에 비해 알고리듬이 간단하고 하드웨어 구현이 용이하여 화상전화, 화상회의, HDTV (High Definition TeleVision) 등에서 동영상 부호화에 사용되고 있다.

BMA에 있어서 가장 대표적인 방법은 탐색 영역내의 모든 점들을 후보점으로 하여 이동벡터를 추정하는 전역탐색 (FS: full search 또는 brute force search)[3] 방법이다. 이 방법은 성능은 우수하나 방대한 양의 계산으로 실시간 처리에 많은 어려움이 따른다. 이러한 단점을 보완하기 위하여 약간의 성능저하는 따르지만 계산량을 줄인 많은 방법이 제안되었다. 이중의 하나가 가산 투영을 이용한 고속 블럭정합 알고리듬[4-7]이다. 2단계 고속 블럭정합 알고리듬은 후보블럭의 왜곡을 측정할 때 2차원 정합을 가산투영을 이용하여 1차원 정합으로 바꾸어 왜곡을 측정함으로써 계산량을 감축시킨다. 그러나, 고속 알고리듬의 개발에도 불구하고 이의 실시간 처리는 힘들기 때문에 병렬 컴퓨터 (parallel computer)를 이용하여 알고리듬의 처리속도를 향상시키는 방법이 연구되고 있다. 최근에는 범용 컴퓨터가 가지는 가격상의 문제, 복잡성 그리고 시스템에 의한 부하 등을 고려하여 특정 알고리듬을 위한 전용 컴퓨터 (special purpose computer)의 개발이 촉진되어 왔는데, VLSI 기술을 이용한 시스톨릭/웨이브프론트 어레이는 그 대표적인 예이다.

시스톨릭 어레이 (systolic array)[8-9]는 VLSI기술을 이용하여 특정한 알고리듬의 수행 속도를 향상시키기 위해 최대한의 동시실행 (concurrency)을 이룬 전용 하드웨어 구조이다. 이의 특징은 모듈성 (modularity), 규칙성 (regularity), 국부적 연결성 (local interconnection), 고도의 종속 연결성 (pipelining), 잘 동기된 다중처리 (multiprocessing) 등이다. 이 구조는 제어가 간단하고 단위시간당 처리량 (throughput)이 많아 DSP (Digital Signal Processing) 분야에 많이 사용된다.

전역탐색 기법은 계산량은 많으나, 성능이 우수하고 이의 강건성 (robustness), 규칙성, 고정단계 연산 (fixed-step operation) 등 하드웨어 구현에 매우 적합한 특성을 가지고 있기 때문에 움직임 추정기의 하드웨어를 위한 알고리듬으로 가장 많이 사용되어 왔다.[10-14] 기존의 연구는 대부분 시스톨릭 어레이를 이용하였으며, 그 중에서도 SMCA (Systolic Mesh Connected Array)가 주종을 이루고, 여러 구조가 있지만 기본적으로는 Komarek과 Pirsch[11]의 구조를 크게 벗어나지 않는다. 시스톨릭 어레이 구조는 처리량을 높이기 위해 동시에 여러 개의 메모리를 참조하는데, 단위시간에 참조할 수 있는 메모리의 양은 제한되어 있으므로 메모리 대역폭의 문제가 발생할 수 있다. Hsieh와 Lin[13]은 이를 보완하기 위하여 Komarek과 Pirsch의 한가지 구조에 쉬프트 레지스터를 추가하여 대역폭의 문제를 해결하려 하였다. SMCA구조는 데이타의 올바른 전파를 위해 입력 데이타가 순차적으로 지연되어 입력되는 입력 skew 현상이 발생하는데 이는 데이타 latency의 증가를 가져온다. 이를 보완하기 위하여 트리 구조로 어레이를 구성한 Jehng 등[14]의 방법도 있다. 이 방법은 처리량도 높고 블럭간의 유사성을 이용하지 않기 때문에 TSS (Three Step Search) 등 다른 알고리듬에도 적합한 장점이 있으나, 다른 방법에 비해 단위 시간당 참조하는 메모리의 양이 월등히 많은 단점이 있다.

본 연구는 1, 2차원 시스톨릭 어레이, 완전 트리, 그리고 16 cut 트리 구조를 이용한 4종류의 전역탐색 움직임 추정기와 가산투영을 이용한 2단계 고속 블럭정합 알고리듬[6-7]의 하드웨어 구현에 중점을 두고 수행하였다. 본 논문의 구성은 Ⅱ장에서 전역탐색 블럭정합 알고리듬과 가산투영을 이용한 2단계 고속 블럭정합 알고리듬을 서술하고, Ⅲ장에서 제안된 하드웨어 구조를 보였다. Ⅳ장에서 SPW (Signal Processing Worksystem)[15]와 VHDL (Very high speed integrated circuits Hardware Description Language)[16]을 이용한 모델링 및 시뮬레이션 결과를 보이고, Ⅴ장에서 이의 성능을 분석하고, 마지막으로 Ⅵ장에서 결론을 맺는다.

Ⅱ. 블럭정합 알고리듬

블럭정합 알고리듬은 시간적으로 서로 이웃한 두 프레임에서 각 프레임을 일정한 크기의 블럭으로 나눈 후 해당 블럭의 움직임을 추정하는 알고리듬이다. 최대 변위가 p이고 부블럭의 크기가 $N\times N$일 때 움직임 추정은 $(k-1)$번째 프레임에서 $(N+2p)$ $(N$

+ 2p)의 탐색영역을 정하고 k번째 프레임에서의 $N \times N$ 블럭과 같은 크기의 블럭을 (k - 1)번째 프레임에서 탐색영역을 벗어나지 않도록 하여 서로간의 유사도를 계산하여 최적의 블럭을 찾아 이때의 변위를 계산하여 해당 블럭의 움직임 정도를 구한다. 유사도를 측정하는 평가 함수로는 MSE (Mean Square Error)나 MAD (Mean Absolute Difference)를 주로 사용한다.

블럭정합 알고리듬에는 탐색영역 내의 모든 점들에 대해 움직임을 추정하는 전역탐색 방식과 이에 비해 성능은 떨어지지만 계산시간에서 이득을 얻을 수 있는 TSS 방식, DMD 방식, menu vector 방식, OTS 방식, cross search 방식, 그리고 계층적 블럭정합 알고리듬 등이 있다.[1,3] 이들은 탐색점 수를 줄여 계산량을 감축하는 방법인데 비하여 가산투영을 이용한 2단계 고속 블럭정합 알고리듬[6,7]은 탐색점의 수는 원래의 값을 사용하고 블럭들의 유사도 검사시의 계산량을 줄이는 방법이다. 유사도 측정 계산량을 줄이기 위해 두단계의 정합과정을 사용하였다. 첫번째 정합단계에서는 블럭간의 2차원 정합계산을 가산투영을 이용하여 1차원 정합으로 변환하여 정합 계산량을 감축하였다. 그러나 2차원 정합을 1차원 정합으로 변환할 때 발생하는 정보 손실로 인한 오정합을 보상하기 위해 두번째 정합과정에서 2차원 정합을 사용하여 부정확한 정합으로 인한 화질저하를 최소화하였다.

가산투영을 이용한 2단계 고속 블럭정합 과정을 설명하면 다음과 같다.

과정 1) 현재 영상에 대해 수평, 수직 가산투영값을 계산한다.

과정 2) 이전 영상에 대한 수평, 수직 가산투영값을 계산한다.

과정 3) 1차원 가산투영값들을 사용하여 현재 영상과 이전 영상사이의 왜곡함수값을 계산한다.

과정 4) 위와 같은 과정을 통하여 구한 모든 탐색점의 왜곡함수값중 최소값을 갖는 정합점을 찾는다.

과정 5) 과정 4에 의해 얻어진 한개의 탐색점과 주변 8개의 탐색점에 대해 2차원 왜곡함수값을 계산한다.

과정 6) 과정 5의 왜곡함수값중 최소값을 선택하여 최종 정합점을 결정한다.

Ⅲ. 가산 투영을 이용한 2단계 고속 블럭정합 알고리듬의 하드웨어 구조

기존의 움직임 추정기 구조[10-14]는 Komarek와 Pirsch[11]의 SMCA 구조와 Jehng 등[14]의 트리 구조를 사용하였다. 마찬가지로 본 장에서 설명할 제안한 가산투영을 이용한 2단계 고속 블럭정합 알고리듬의 하드웨어 구조도 시스톨릭 어레이를 이용하여 구현하였다.

설계기법 설명시 편의를 위하여 최대 변위 p를 ±3으로 하고 부블럭의 크기는 4×4로 하여 1차원 정합과정 블럭, 2차원 정합과정 블럭 및 부화소 움직임 추정블럭에 대하여 설명한다.

가산투영을 이용한 2단계 고속 블럭 정합 알고리듬[6-7]의 아키텍쳐는 수평, 수직 가산투영값을 이용한 1차원 정합과정과 여기서 얻어진 한개의 탐색점을 중심으로 주변 8개의 탐색점들과 중심점에 대해 2차원 왜곡함수값을 이용하여 정합점을 찾는 2차원 정합과정의 2개의 부분으로 구성하였다. 그리고 보다 더 정확한 움직임을 검출하기 위해 부화소 단위 움직임 검출 블럭을 추가하였다.

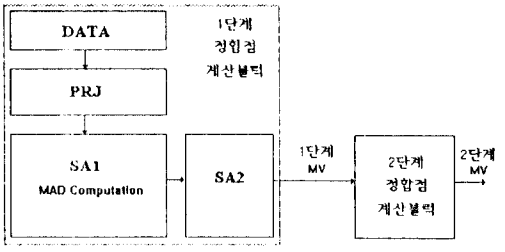


그림 1. 2단계 BMA의 블럭도

Fig. 1. Block diagram of the first stage of the two-stage BMA.

제안한 가산투영을 이용한 2단계 고속 블럭 정합 알고리듬의 아키텍쳐를 그림 1에 나타내었다. 1차원 정합과정은 영상의 화소값을 출력하는 DATA 블럭, 수평과 수직 가산투영값을 계산하는 PRJ 블럭, PRJ 블럭에서 구한 수평, 수직의 가산투영값을 이용하여 현재영상과 이전영상 블럭 사이의 왜곡함수값을 계산하는 SA1 블럭, 그리고 SA1 블럭에서 구한 왜곡 함수값으로부터 최소의 왜곡함수값을 찾는 SA2 블럭으로 구성하였다. 그리고, 2차원 정합과정은 Komarek와 Pirsch[11]의 움직임 추정기 구조중 AB1 구조와 Jehng 등[14]의 트리 구조를 사용하였다. 위의 과정을 거쳐 2단계 움직임 벡터 즉, 최종 정합점의 위치값이 출력된다.

DATA 블럭은 현재영상과 이전영상을 구조에 맞는 순서대로 수평, 수직의 가산투영값을 계산하는 블럭의 입력으로 보낸다. 수평, 수직 가산투영값을 계산

하는 PRJ 블럭 (V0 V9, H0 H9)의 입력으로 보내는 순서를 그림 2에 나타내었다.

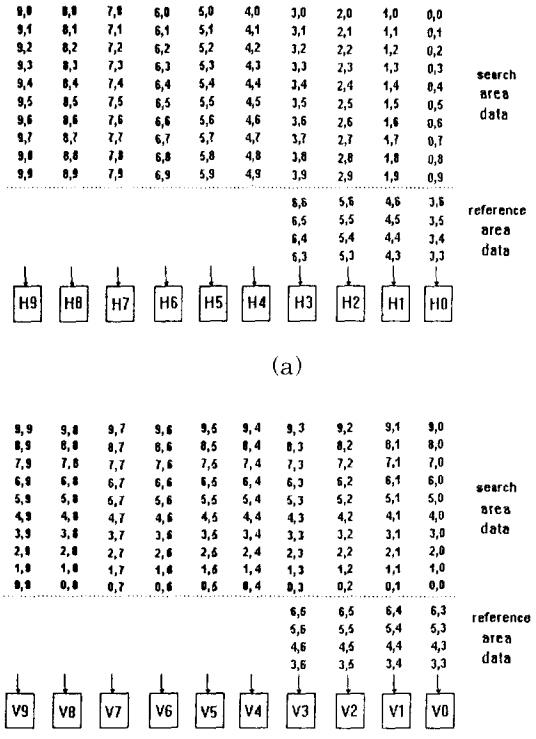


그림 2. PRJ 블럭의 입력 (a) 수평 가산투영 블럭의 입력 (b) 수직 가산투영 블럭의 입력

Fig. 2. Input to a PRJ block.
(a) Horizontal projections
(b) Vertical projections.

PRJ 블럭은 n개의 값을 더하여 출력한 다음, n개의 값중 맨처음 입력으로 들어온 것을 제외하고 n-1개의 값의 합은 계속적으로 필요하므로 본 구조에서는 가장 먼저 받아들였던 입력은 빼고 새로 받아들인 입력을 n-1개의 합과 더하여 출력한다. PRJ 블럭중에서 MAD를 계산하는 부분의 입력으로 쓰이는 부분은 V0 V3과 H0 H3이고 현재 사용하지 않는 가산투영값을 계속적으로 받아들여 출력시키는 부분은 V4 V9와 H4 H9이다.

4개씩의 수평, 수직 가산투영값을 입력으로 하여 블럭의 왜곡 함수값을 구하는 SA1 블럭은 4×4 블럭 7개의 왜곡함수값을 계산하는 부분과 현재 사용하지 않는 가산투영값을 계속적으로 받아들여 왜곡함수값을 계산하는 부분으로 전파시키는 부분으로 구성되어 있는데 이 구조가 동작하는 과정은 다음과 같다. 기존의 방법중에는 현재영상의 가산투영값을 PE (processing element)안에 저장하였다고 가정하고 구조를 구성할 수 있는데 이 구조는 현재영상의 가산투영값을 입력으로 받아들이도록 설계되었다. 현재영상의 가산투영값이 입력으로 들어올 때와 이전영상의 가산투영값이 입력으로 들어올 때의 동작과정을 따로 설명하면 다음과 같다.

현재영상의 가산투영값이 입력으로 들어올 때는 그림 3에서 R로 표시된 부분은 1차원 정합과정시 필요한 과정을 수행하지 않고 M으로 표시된 부분만 현재영상의 수평, 수직 가산투영값을 한 행 4개의 M에 저장하는 과정을 수행한다. 즉 하나의 M에 1개씩의 수평, 수직 가산투영값을 저장한다. M의 동작과정은 위에서 아래로, 입력을 한 클럭후에 출력으로 내보내면서 M안에 입력을 저장한다. 이러한 과정을 계속적으로 수행하면 같은 열의 M에는 같은 값이 저장된다.

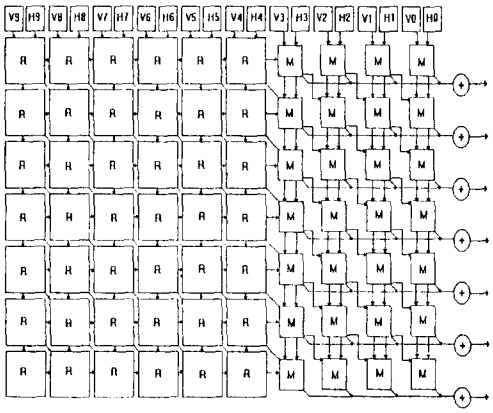


그림 3. 왜곡값 계산 블럭도

Fig. 3. Block diagram for distortion calculation.

M에 현재영상의 가산투영값이 저장되어 있으면서 이전영상의 가산투영값이 입력으로 들어올 때의 동작과정은 수평 가산투영값은 M과 R에서 맨 밑의 행에 전파될 때까지는 수직 방향으로 전파시키고 그 다음 클럭부터는 수평방향으로 전파시킨다. 수직 가산투영값은 M과 R에서 모두 대각선 방향으로 전파시킨다. 이러한 각각의 PE의 전파흐름 과정을 수행하면 그림 3의 7개 행에 현재영상의 수평, 수직 가산투영값이 저장되어 있으므로 한번의 클럭으로 이전영상의 4×4 블럭 7개에 대해서 왜곡함수값을 구할 수 있다. 그러

므로 첫번째 왜곡함수값이 출력된 후 7 클럭이 소요되면 이전영상 전체블럭에 대한 왜곡함수값을 구할 수 있다.

위의 과정에서 구한 PRJ 블럭의 출력인 블럭의 왜곡함수값을 입력으로 하여 1차원 정합과정의 최종정합점을 구하는 부분은 그림 4에 도시하였다. 그림에서 처음과 마지막 입력 사이에는 6 클럭의 차이가 있으므로 마지막 PE인 D에 클럭을 맞추기 위하여 맨 앞단의 C에 지연을 취하였다. C에 지연을 취하는 것은 맨 앞단의 C뿐이다. 전체적으로 클럭을 맞추어 왜곡함수의 최소값을 구하는 과정을 마지막 왜곡함수값의 최소값을 구하는데까지 수행하여 이전영상 전체블럭 중 최소 왜곡함수값을 구한다. 여기서 6개의 입력은 C 블럭에 입력되어 왜곡함수의 최소값을 구하는 과정을 수행하고 마지막 1개는 D 블럭에 입력되어 단순히 한 클럭을 지연시켜 C의 입력으로 출력한다. 위의 과정을 수행하면 한 클럭에 4×4 블럭 7개에 대한 왜곡함수값의 최소값이 출력되는데 그 이후로 계속적으로 출력되는 7개의 최소값중 다시 최소값을 찾는 부분은 그림 4에 MIN으로 표시되어 있다. MIN의 출력값이 1차원 정합과정의 최종출력으로 최소 왜곡함수값을 갖는 블럭의 위치값을 출력한다.

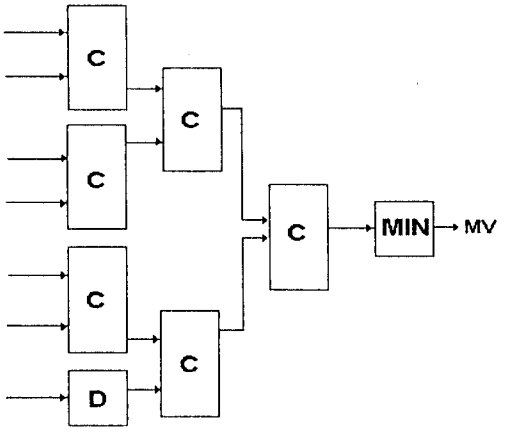


그림 4. 움직임 벡터 추출 블럭도
Fig. 4. Block diagram for motion vector detection.

1차원 정합과정에서 사용한 PE들을 그림 5에 도시하였다. 그림 5(a)는 R 블럭으로 현재영상의 가산투영값이 입력으로 들어오는 경우에는 1차원 정합과정시 필요한 과정을 수행하지 않는다. 이전영상이 입력으로 들어올 때에는, 수평 가산투영값일 경우에는 맨 밑의 행에 전파될 때까지는 수직 방향으로 전파시키고, 그 다음 클럭부터는 수평방향으로 전파시킨다. 수직 가산투영값은 대각선으로 전파시킨다. 그림 5(b)는 M 블럭으로서 현재영상의 가산투영값이 입력으로 들어오는 경우에는 수평, 수직 가산투영값을 수직방향으로 전파시키고, 이전 영상이 입력으로 들어올 때에는 R 블럭과 같이 수평 가산투영값일 경우에는 맨 밑의 행에 전파될 때까지는 수직 방향으로 전파시키고 그 다음 클럭부터는 수평방향으로 전파시킨다. 그리고, 수직 가산투영값은 R 블럭과 마찬가지로 대각선으로 전파시킨다. 그림 5(c)는 두개의 입력중 작은 값을 출력으로 내보내는 C로 두개의 입력을 받아들여 작은 값을 출력으로 내보낸다. 입출력이 표시된 수식에서 min(a,b)는 a, b중 작은 값을 의미한다. 그림 5(d)는 D 블럭으로 지연을 행하는 과정으로 단순히 한 클럭 후에 입력을 출력으로 내보내는 과정을 수행한다. 그림 5(e)는 MIN 블럭으로서 7개의 입력을 순서적으로 받아들여 최소값을 구해 1차원 최종 정합점을 구하는 블럭이다. 입출력이 표시된 수식에서 MIN(mi)는 입력중 구한 최소값을 나타낸다.

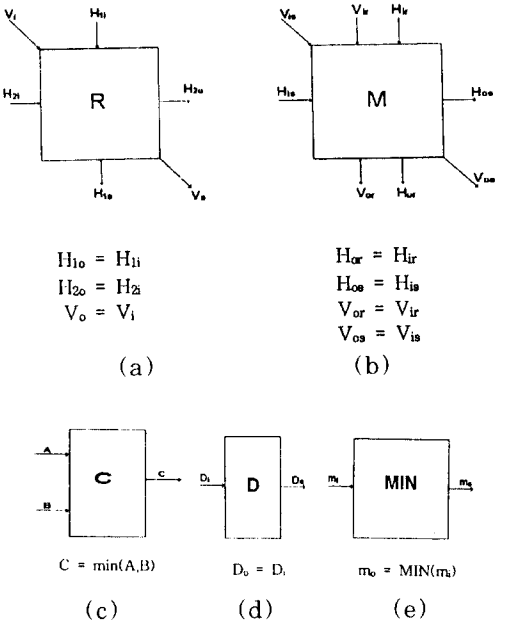


그림 5. 각 PE의 기능
(a) R 블럭 (b) M 블럭 (c) C 블럭
(d) D 블럭 (e) MIN 블럭
Fig. 5. Function definition of each PE.
(a) R block (b) M block (c) C block
(d) Dblock, (e) MIN block.

앞에서 최종적으로 구한 1차원 정합점을 중심으로 주변 8개의 탐색점들과 중심점에 대해 2차원 왜곡함

수값을 구하여 왜곡함수값중 최소값을 선택해야 하는데 이 구조는 Komarek와 Pirsch의 AB1 구조와 Jehng 등의 트리 구조를 사용하였고 부화소 단위 움직임 검출 블럭도 위의 구조를 약간 수정한 후 사용하였다. 부화소 단위 움직임 추정 구조에서 수정한 부분은 현재 영상과 이전 영상의 값을 구조에 맞는 순서대로 출력시키기 위한 부분과 왜곡함수값의 최소값을 구하는 부분이다. 부화소 단위 움직임 추정 블럭에서는 이전영상을 구조에 적합한 순서대로 출력해야 하는데 2차원 정합과정에서는 1차원 최종 정합점과 그 주위의 8점에 대하여 구조에 적합한 순서대로 출력하면 된다. 그리고 왜곡함수값의 최소값을 구하는 부분에서는 부화소의 위치대신에 최종 정합점과 주위의 8점에 대한 위치값을 출력으로 보내면 된다.

Ⅳ. 블럭정합 알고리듬의 하드웨어 구현

본 장에서는 기존의 블럭정합 알고리듬과 가산투영을 이용한 2단계 고속 블럭정합 알고리듬의 SPW와 VHDL 모델링 및 시뮬레이션 결과에 대하여 설명한다. 부블럭의 크기 N은 16, 최대 변위 p를 ±7로 하여 실험하였다.

전역탐색 움직임 추정기는 1, 2차원 시스톨릭 어레이 구조인 Komarek와 Pirsch의 AB1, AB2 구조와 Jehng 등의 완전 트리 구조, 16 cut 트리 구조를 SPW와 VHDL을 이용하여 구현하였다. 그림 6은 움직임 추정기의 전체 시스팀을 나타낸다.

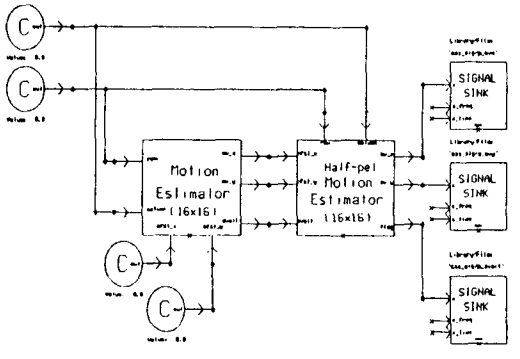


그림 6. 움직임 추정기 시스팀
Fig. 6. Motion estimation system.

1차원 시스톨릭 어레이 구조로는 Komarek와 Pirsch의 AB1 타입을 16×16 블럭에 맞게 확장한 것을 시뮬레이션하였다. 이 구조는 각 화소들간의 절대차 (absolute pixel difference)를 계산하고 이를 누적하는 N개의 AD 모듈이 직렬로 있고, 이의 양쪽에서 차이가 계산되어야 할 두개의 테이타 입력 발생기가 있다. AD 어레이의 최종 출력단에 정합점을 찾는 시스팀 콘트롤러가 있다. 그림 7은 AB1 타입 구조를 SPW를 이용하여 모델링한 시스팀이다. 여기서 AD 어레이는 4개의 AD로 이루어진 블럭이다.

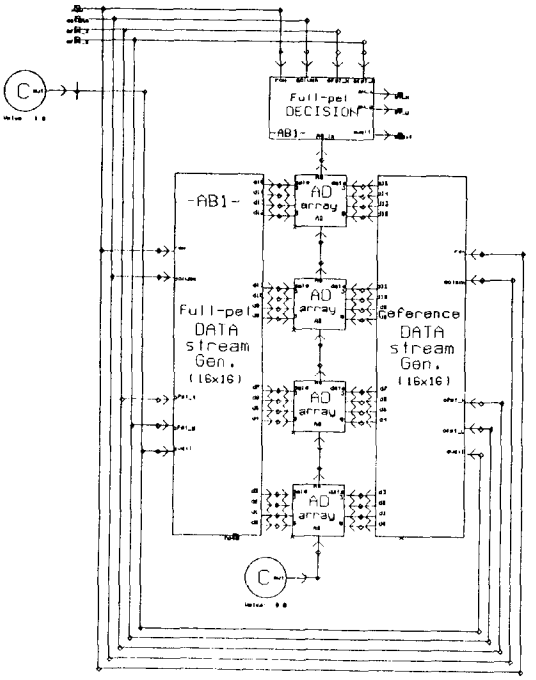


그림 7. ABI 타입 움직임 추정기
Fig. 7. ABI type motion estimator.

그림 8은 2차원 시스톨릭 어레이 구조인 AB2 구조를 SPW로 모델링한 시스팀이다. AB1의 경우와 같이 시스톨릭 어레이 부분은 AD로 이루어져 있다. 그외 어레이부의 입력 데이타를 발생시키는 블럭과 절대차를 누적시키는 덧셈기 어레이, 그리고 정합점을 찾는 블럭으로 이루어져 있다. 여기서 PE 어레이와 ADDER 어레이블럭은 각각 4×4개의 PE와 4개의 덧셈기로 이루어진 블럭들이다.

완전 트리 구조는 메모리 대역폭의 문제가 심각하기 때문에 현재의 반도체 기술 수준으로는 어려움이 따른다. 그림 9는 16-cut 트리 구조를 SPW를 이용하여 모델링한 시스팀이다. 16-cut 트리 구조에는 완전 트리 구조에서 결정 블럭 앞에 16개의 부분 왜곡값을 누적시키는 누산기 (accumulator)와 화소의 절대차를 내는 D의 입력 부분에 움직임 블럭의 값을 번갈아 넣어주는 스위치와 레지스터 어레이가 추가로 필요하다. 완전 트리 구조는 그림에서 누산기가 제거

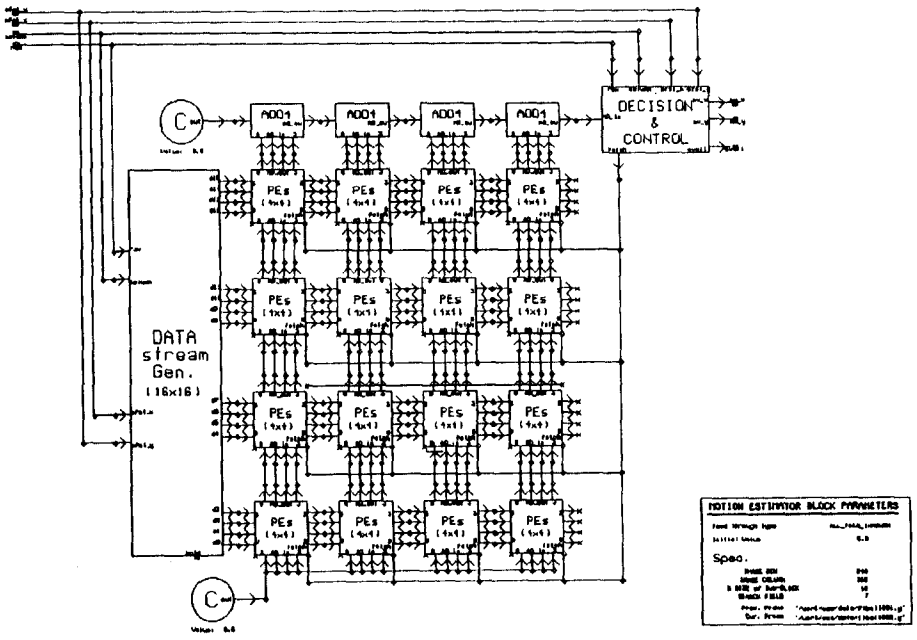


그림 8. AB2 움직임 추정기
Fig. 8. AB2 type motion estimator.

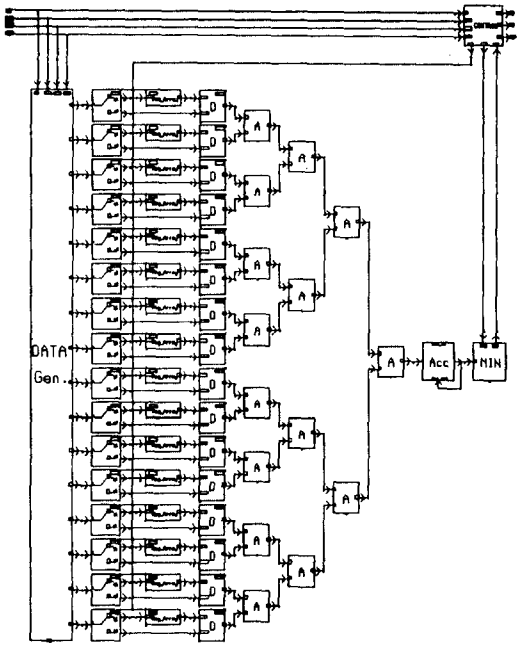


그림 9. 트리 구조 움직임 추정기
Fig. 9. Tree structured motion estimator.

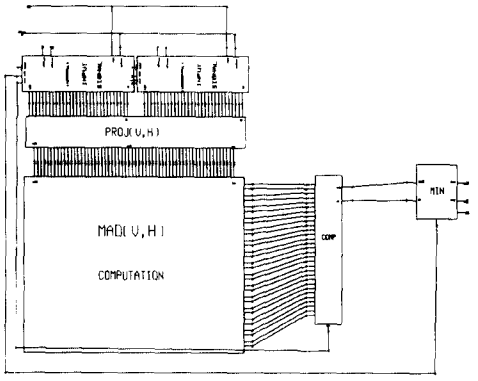
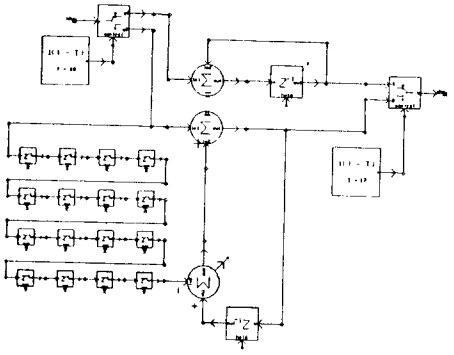


그림 10. 그림 2단계 BMA의 1단계 SPW 블럭도
Fig. 10. SPW block diagram of the first stage of the two-stage BMA.

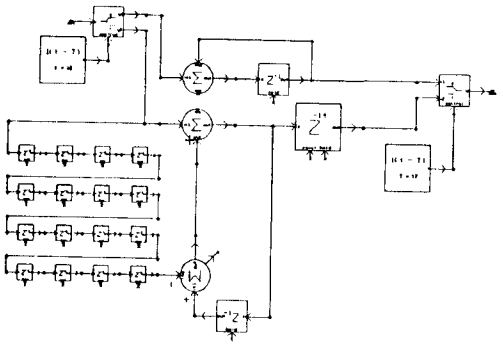
되고 16개의 D 대신에 17개의 절대차를 빼고 하나로 합하는 tree를 넣어 주어야 한다. 이렇게 256 화소의 절대차를 내고 이를 더해주어 왜곡값으로 사용한다.

가산투영을 이용한 2단계 고속 블럭정합 알고리듬의 SPW 모델링은 1차원 정합하는 과정과 2차원 정합하는 과정으로 구현되어 있다. 그림 10은 1차원 정합 과정 전체블럭으로 현재 영상과 이전영상을 읽어 제안한 구조가 원하는 순서대로 출력하는 부분과 수

평, 수직의 가산투영값을 계산하는 부분, 그리고 MAD 계산하는 부분 및 MAD에서 계산된 출력값중 최소값을 찾는 부분으로 구성되어 있다. 2차원 정합 과정은 전역탐색 블럭정합 알고리듬에서 최대 변위 p가 ±1인 경우이므로 그림 8의 AB1 타입과 그림 9의 트리 구조를 N은 16이고 p는 ±1로 하여 구현하였다.



(a)



(b)

그림 11. 가산투영값 계산 SPW 블럭
(a) 수평 가산투영 (b) 수직 가산투영
Fig. 11. SPW block diagram for calculation ointegral projections. (a) Horizontal projections, (b) Vertical projections.

그림 11(a)는 수평 가산투영값을 계산하는 블럭 즉, 앞에서 설명한 H 블럭으로 입력을 순서적으로 받아들여 이전까지 합해진 결과에서 가장 먼저 받아들인 입력을 제외하고 새로이 받아들인 입력을 더하여 출력으로 내보낸다. 이러한 과정을 계속적으로 수행함으로써 현재영상의 올바른 가산투영값이 16번째 클럭에서 출력된다. 그리고 그 이후 16번째 클럭부터 이전영상의 올바른 값이 출력되기 시작한다. 그림 11(b)는 수직 가산투영값을 계산하는 블럭 즉, V 블럭으로 수평 가산투영값을 계산하는 블럭과 같다. 수평 가산투영값을 계산하는 블럭과의 차이점은 현재영상의 가산투영값을 계산하는 과정은 같지만 이전 영상의 가산투영값을 계산할 때에는 수평 가산투영값이 MAD를 계산하는 부분의 맨 마지막 행에 전파되고부터 수직 가산투영값을 출력하기 시작한다.

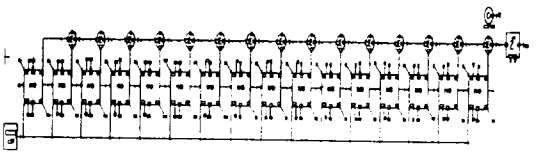


그림 12. 16×16 부블럭의 MAD 계산 SPW 블럭
Fig. 12. SPW block diagram for MAD calculation of a 16×16 subblock.

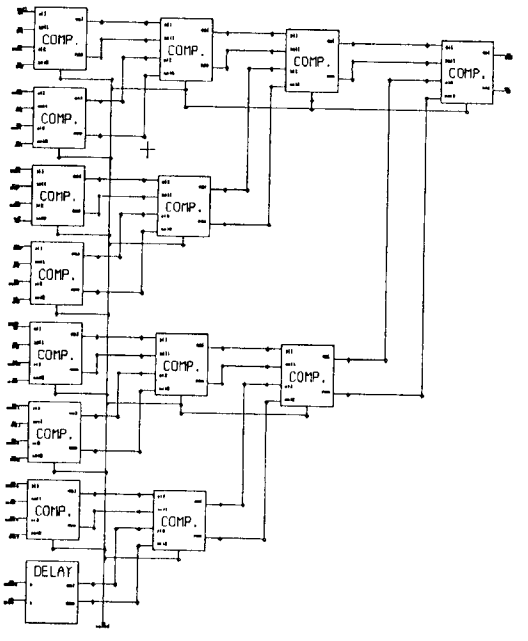


그림 13. 최소 왜곡함수값을 구하는 SPW 블럭
Fig. 13. SPW block diagram for MAD calculation minimum distortion.

그림 12는 16×16 블럭의 MAD를 계산하는 블럭이다. 그림 13은 MAD 블럭에서 구한 왜곡 함수값

으로부터 가장 작은 왜곡함수값을 찾는 블럭으로 그림 4의 블럭을 SPW로 구현한 것이다.

2차원 정합 과정은 전역 탐색 움직임 추정기로 개발된 구조중 Komarek와 Pirsch의 AB1 구조 (그림 7)와 Jehng 등의 완전 트리와 16 cut 트리 구조 (그림 9)를 이용하여 구현하였다.

지금까지 전역탐색 블럭정합 알고리듬과 가산투영을 이용한 2단계 고속 블럭정합 알고리듬의 SPW 모델링에 대하여 설명하였으며 부화소 단위 움직임 검출블럭은 앞에서도 언급하였듯이 AB1 구조와 트리 구조를 이용하여 구현하였다. AB1 구조와 트리 구조의 부화소 움직임 추정기의 전체 구성은 각각 그림 7, 9와 같고 AB1 타입에서의 화소단위 움직임 추정기와의 차이점은 탐색 영역의 블럭값을 PE의 입력으로 넣어주는 탐색 영역 데이타 입력 발생기 (search area data generator)와 시스팀 콘트롤러의 동작에 있다. 트리 구조에서는 데이타 입력 발생기와 콘트롤러만 다르게 구현하였다. 그림 14는 움직임 추정기를 SPW로 모델링하고 시뮬레이션한 결과이다.

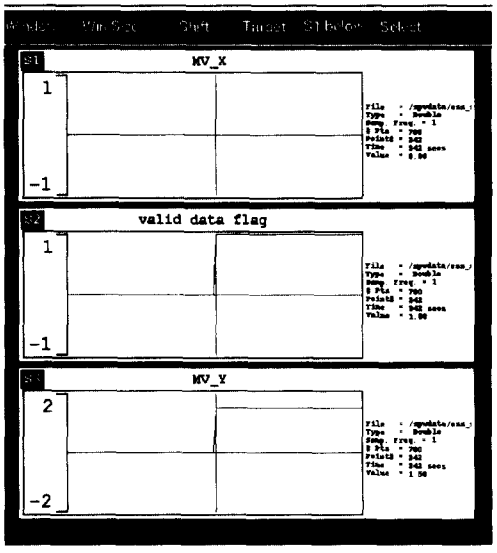


그림 14. SPW 시뮬레이션 결과
Fig. 14. SPW simulation results.

위에서 설명한 모든 구조, 즉 전역 탐색 움직임 추정기의 AB1, AB2 타입, 완전 트리 구조 및 16 tree 트리 구조 그리고, 가산 투영을 이용한 2단계 고속 블럭 정합 알고리듬의 1, 2차원 정합 과정 및 AB1 구조와 트리 구조를 이용한 부화소 단위 움직임 검출 블럭을 Vantage™ 사의 VHDL [16] 을 이용하여 각각의 PE 등 최소단위의 블럭은 행위 단계 (behavioral level)로 블럭을 기술하고 이들을 구조 단계 (structual level)로 기술하여 최종 움직임 추정기를 구성하였다. 그림 15는 움직임 추정기를 Vantage™ 사의 VHDL 시뮬레이터인 VSS (Vantage Spread Sheet)를 이용하여 VHDL로 모델링하고 시뮬레이션한 결과이다.

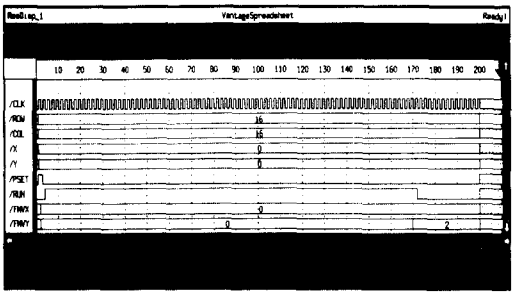


그림 15. VSS를 이용한 VHDL 시뮬레이션 결과
Fig. 15. VHDL simulation results using VSS.

V. 성능 분석

본 장에서는 시스톨릭 어레이를 이용하여 구현한 전역 탐색 움직임 추정기와 가산투영을 이용한 2단계 고속 블럭 정합 알고리듬의 하드웨어의 성능을 분석한다.

$N \times N$ 탐색 블럭의 최대 변위를 p로 했을 때 제안한 가산투영을 이용한 2단계 고속 블럭정합 알고리듬의 시스톨릭 어레이 구조중 1차원 정합과정에 소요되는 processing time은 $3N + 4p + \log_2 N$ 이다. 표 1은 각 방법의 소요 시간을 processing time으로 나타낸 것이다. 그리고 N이 16, p가 7인 경우를 나타내었다. 각 PE에서 소요되는 시간을 1 processing time으로 생각한다. 표 1에서 고속 알고리듬의 경우 괄호안은 2차 정합의 구조이다. 표 1(a)를 보면 2단계 고속 알고리듬의 경우 1차원 정합과정은 제안한 구조를 사용하고 2차원 정합과정은 AB1을 사용하였을 때의 소요 시간은 240 processing time이다. 그러나, 전역탐색 움직임 추정기를 AB1으로 구현하였을 때의 소요시간은 3617 processing time이 소요된다. 그러므로 약 15배 이상 빠르게 동작함을 알 수 있다. 그리고 전역탐색 움직임 추정기중 가장 빠르게 동작하는 완전트리의 소요시간이 242 processing time이고 2단계 고속 알고리듬의 경우 소요 시간이 106 processing time이므로 2배 이상 빠른 것을 알

수 있다. 그러므로 가산투영을 이용한 고속 알고리듬을 하드웨어로 구현한 것이 표에서도 알 수 있듯이 15배에서 2배까지 빠르게 동작함을 확인할 수 있다. 표 1(b)는 부화소 단위 움직임 추정기에 사용된 구조들의 소요시간을 나타내었다.

표 1. 각 구조의 소요 시간 비교
(단위 : pro-cessing time)
(a) 화소 단위 탐색
(b) 부화소 단위 탐색

Table 1. Comparison of processing time.
(a) Full-pel search
(b) Half-pel search.

(a)

	소요 시간
AB1	3617
AB2	496
완전 트리	242
16-cut 트리	3607
고속 (AB1)	80 + 160
고속 (완전 트리)	80 + 26
고속 (16-cut 트리)	80 + 151

(b)

	소요 시간
AB1	160
완전 트리	26
16-cut 트리	151

모든 구조의 소요 시간에 있어 블럭의 크기보다는 탐색 영역의 크기가 더욱 크게 영향을 미친다. 그러므로, 탐색 영역이 크면 클수록 고속 알고리듬의 효율은 더욱 증가한다고 말할 수 있다. 최근 움직임 보상 부호화 표준화 동향이 정확한 움직임을 찾기 위해 탐색 영역의 크기를 증가시키는 쪽으로 나아가는 것으로 볼 때 이는 매우 바람직하다고 할 수 있다.

표 2. 필요한 덧셈기의 수 비교

Table 2. Comparison of the number of adders required.

	비용
AB1	$2N + 1 + 1$
AB2	$2N^2 + N + 1$
완전 트리	$N^2 + 2 \log_2 N$
16-cut 트리	$N^2/16 + 2 \log_2 16 + 1$
고속 (1차 정합)	$4N(p + 1) + 2 \log_2 (2p + 1) - 1$

표 2는 데이타를 메모리로부터 읽어오는 부분과 시스템 콘트롤러를 제외하고 어레이 프로세서 부분의 덧셈기의 수를 나타낸 것이다. 여기서 N은 블럭의 크기이고 p는 최대 변위이다. 그리고, 절대차를 구하는 부분과 최소 왜곡을 구하는 블럭은 하나의 덧셈기로 계산하였다. 표 2에서 고속 알고리듬의 경우 2차 정합에서의 덧셈기 수가 빠진 것이다. 표 2를 보면 고속 알고리듬의 경우 2차 정합을 위한 덧셈기가 추가되어야 하므로, 기존의 전역탐색 기법을 위한 구조에 1차정합을 위한 구조가 추가로 들어가는 셈이 된다. $N = 16$, $p = 7$일 때 하드웨어 비용을 비교해 보면, AB2의 경우는 덧셈기의 개수가 529개이고 완전 트리의 경우는 264개인데 비해 2단계 고속 알고리듬의 경우 추가로 들어가는 비용이 519개이므로, 2-3배의 덧셈기가 더 드는 셈이다.

Ⅵ. 결론

본 논문에서는 블럭정합 알고리듬의 하드웨어 설계 기술에 대하여 고찰하였다. 전역탐색 움직임 추정기는 Komarek와 Pirsch의 AB1, AB2 구조와 Jehng 등의 트리구조로 구현하였다. 그리고 가산투영을 이용한 2단계 고속 블럭정합 알고리듬은 제안한 구조와 AB1 구조 및 트리구조로 구현하였다.

블럭 정합 알고리듬을 위한 하드웨어 구조는 비교적 효과적으로 구성되어 있다. 그러나, 알고리듬 자체가 워낙 많은 계산량을 가지기 때문에 하드웨어로 구현하여 실용화하기엔 아직까지는 여러 어려움이 따르는 것이 사실이다. 본 논문에서는 이것을 보완하고자 왜곡 측정 함수를 달리해 계산량을 감축시킨 가산투영을 이용한 2단계 고속 블럭정합 알고리듬을 시스톨릭 어레이를 이용하여 구성한 하드웨어 구조를 제안하였다. 시뮬레이션 결과 가산투영을 이용한 고속 알고리듬을 하드웨어로 구현한 것이 매우 빠르게 효율적으로 동작함을 확인할 수 있었다.

지금까지 이를 주제로 많은 연구가 진행되어 온 것이 사실이지만 영상 처리 분야에서 근본적으로 대두되는 계산량이나 메모리 대역폭 등의 문제로 아직까지 실용화되어 있지 못한 것도 사실이다. 전역탐색 블럭정합 알고리듬은 구조에 따라 메모리 대역폭이나 계산시간의 문제가 내재해 있고, 2단계 고속 알고리듬도 알고리듬 자체의 계산량 감축에서 오는 성능저하와 가산투영 등을 계산하기 위한 하드웨어 비용이 추가되는 등 아직은 연구가 미미한 단계이다. 그러므로, 실시간 처리가 가능하고 하드웨어 구현이 용이한 알고리듬의 개발 및 이의 하드웨어 구현에 관한 연구가 지속되어야 할 것이다.

參 考 文 獻

[1] H. G. Musmann et al., "Advances in picture coding," *Proc. IEEE*, vol. 73, pp. 523-548, Apr. 1985.

[2] A. N. Netravali and J. D. Robbins, "Motion-compensated television coding: Part I," *Bell Syst. Tech. J.*, vol. 58, pp. 631-670, Mar. 1979.

[3] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.*, vol. COM-29, pp. 1799-1808, Dec. 1981.

[4] J.-S. Kim and R.-H. Park, "Feature based block matching algorithm using integral projections," *IEE Electron. Lett.*, vol. 25, pp. 29-30, Jan. 1989.

[5] J.-S. Kim and R.-H. Park, "A fast feature based block matching algorithm using integral projections," *IEEE Journ. Selected Areas Commun.*, vol. SAC-10, pp. 968-971, June 1992.

[6] 김 준식, 박 래홍, 이 병욱, "가산 투영을 이용한 2단계 고속 블럭정합 알고리듬," 전자공학회 논문지 B, 제 30권, 제 1호, pp. 45-55, 1993년 1월.

[7] J.-S. Kim, R.-H. Park, and B. U. Lee, "Two-stage fast block matching algorithm using integral projections," *Journ. Visual Commun. Image Representation*, vol. 4, pp. 336-348, Dec. 1993.

[8] H. T. Kung, "Why systolic architecture?," *IEEE Computer*, vol. 15, pp. 37-46, Jan. 1982.

[9] S. Y. Kung, *VLSI Array Processors*, Prentice-Hall, Englewood Cliffs, NJ, 1988.

[10] K. M. Yang, L. Wu, H. Chong, and M. T. Sun, "VLSI implementation of motion compensation full search block matching algorithm," in *Proc. SPIE Visual Communications and Image Processing '88. Cambridge*, MA, vol. 1001, pp. 892-899, Nov. 1988.

[11] T. Komarek and P. Pirsch, "Array architectures for block matching algorithms," *IEEE Trans. Circuits Syst.*, vol. CAS-36, pp. 1301-1308, Oct. 1989.

[12] L. D. Vos and M. Stegherr, "Parameterizable VLSI architecture for the full search block matching algorithm," *IEEE Trans. Circuits Syst.*, vol. CAS-36, pp. 1309-1316, Oct. 1989.

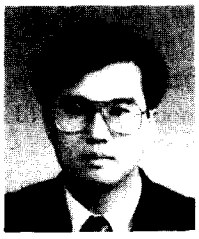
[13] C.-H. Hsieh and T.-P. Lin, "VLSI architecture for block matching motion estimation algorithms," *IEEE Trans. Circuits Syst. for Video Tech.*, vol. 2, pp. 169-175, June 1992.

[14] Y.-S. Jehng, L.-G. Chen, and T.-D. Chiueh, "An efficient and simple VLSI tree architecture for motion estimation algorithms," *IEEE Trans. Signal Processing*, vol. SP-41, pp. 889-900, Feb. 1993.

[15] *Designer/BDE™ User's* Guide, Comdisco Systems Inc., ver 3.0, 1992.

[16] *VantageSpreadsheet™ User's* Guide, Vantage Systems Inc., 1988.

著者紹介



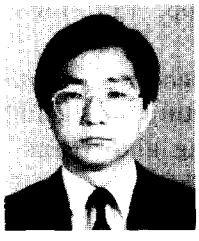
潘 聲 範(準會員)

1967年 11月 1日生. 1991年 2月 서강대학교 전자공학과 졸업 (공학사). 1993年 2月 ~ 현재 서강대학교 대학원 전자공학과 (석사과정). 주관심 분야는 영상 처리, VLSI 설계 등임.

•

金 俊 植(正會員) 第 29卷 B編 第 4號 參照
현재 호서대학교 전자공학과 조교수

•



趙 威 德(正會員)

1958年 11月 17日生. 1981年 2月 서강대학교 전자공학과 졸업 (공학사). 1983年 2月 한국과학기술원 전기 및 전자공학과 졸업 (공학석사). 1987 年 2月 한국과학기술원 전기 및 전자공학과 졸업 (공학박사). 1983年 3月 ~ 1990年 3月 금성전기(주) 기술연구소 디지탈 신호처리 연구실장. 1990年 4月 ~ 1991年 10月 생산기술연구원 HDTV 사업단 조교수. 1991년 11月 ~ 현재 전자부품종합기술연구소 수석연구원 (디지탈이동통신 개발팀장). 주관심 분야는 디지탈 통신시스템, 디지탈 통신 신호처리 기술, 디지탈 이동통신 baseband 신호처리 chipset 설계 등임.

蔡 承 秀(準會員)

1970年 3月 15日生. 1993年 2月 서강대학교 전자공학과 졸업 (공학사). 1993年 2月 ~ 현재 서강대학교 대학원 전자공학과 (석사과정). 주관심 분야는 영상 처리 등임.

•

朴 來 弘(正會員) 第 23卷 第 6號 參照
현재 서강대학교 전자공학과 교수

•

林 信 一(正會員)

1957年 4月 2日生. 1980年 2月 서강대학교 전자공학과 졸업 (공학사). 1983年 2月 서강대학교 대학원 전자공학과 졸업 (공학석사). 1990年 8月 ~ 현재 서강대학교 대학원 전자공학과 (박사과정). 1980年 2月 ~ 1981年 2月 (주)한국소프트웨어 근무. 1982年 3月 ~ 1992年 1月 한국전자통신연구소 선임연구원. 1992년 1月 ~ 현재 전자부품종합기술연구소 주문형반도체설계센터 선임연구원. 주관심 분야는 VLSI 설계, Mixed Mode 회로설계 (ADC/DAC등), Testable Design 등임.