論文96-33B-6-17

시스톨릭 어레이를 이용한 블럭정합 알고리즘의 VLSI 구조

(VLSI Architectures for Block Matching Algorithms Using Systolic Arrays)

潘 聲 範 *, 蔡 承 秀 *, 朴 來 弘 **

(Sung Bum Pan, Seung Soo Chae, and Rae-Hong Park)

요 약

본 논문에서는 전역 탐색 블럭정합 알고리즘과 전역 탐색 블럭정합 알고리즘에 비하여 성능이 비슷하고 계산량을 현저하게 줄인 가산 투영을 이용한 2단계 블럭정합 알고리즘의 VLSI 구조를 제안한다. 제안한 구조는 적은 하드웨어 복잡도로 기존의 구조보다 빠르게 동작한다. 또한 VHDL를 이용하여 모델링하고 시뮬레이션하여 전역 탐색 블럭정합 알고리즘과 2단계 블럭정합 알고리즘을 위한 제안한 구조가 정상적으로 동작함을 확인하였다.

Abstract

In this paper, we propose VLSI architectures for the full search block matching algorithm (FS BMA) and two-stage BMA using integral projections that reduce greatly computational complexity with its performance comparable to that of the FS BMA. The proposed VLSI architectures are faster than the conventional ones with lower hardware complexity. Also the proposed architectures of the FS BMA and two-stage BMA are modeled in VHDL and simulated to show their functional validity.

I. 서 론

동영상 부호화에 있어 많이 사용되는 움직임 보상 부호화 (motion compensated coding)[1-4]는 움직임을 검출하여 이에 따른 시간적인 중복성을 제거하고, 움직임을 나타내는 이동 벡터와 움직임 보상에 따른 예측 오차를 전송하여 영상 데이터를 압축하는 기법이다. 움직임을 검출하는 기법중 블럭정합 알고리즘 (BMA: Block Matching Algorithm)은 다른 방법에 비해 알고리즘이 간단하고 하드웨어 구현이 용이하여 MPEG (Moving Picture Experts Group), HDTV (High Definition TeleVision) 등에서 동영상 부호화에 사용되고 있다.

BMA는 시간적으로 서로 이웃한 두 프레임에서 각 프레임을 일정한 크기의 블럭으로 나눈 후 해당 블럭의 움직임을 추정하는 알고리즘이다. 최대 변위가 p이고 부블럭의 크기가 $N \times N$일 때 움직임 추정은 $(k-1)$번째 프레임에서 $(N+2p) \times (N+2p)$의 탐색 영역을 정하고 k번째 프레임에서의 $N \times N$ 블럭과 같은 크기의 블럭을 $(k-1)$번째 프레임에서 탐색 영역을 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.

BMA에는 탐색 영역 내의 모든 점들에 대해 움직임을 추정하는 전역 탐색 (FS: Full Search) 방식과 이

* 準會員, ** 正會員, 西江大學校 電子工學科

(Department of Electronic Engineering, Sogang Univ.)

接受日字: 1995年10月2日, 수정완료일: 1996年4月24日

에 비해 성능은 떨어지지만 계산 시간에서 이득을 얻을 수 있는 TSS (Three Step Search) 방식, DMD (Direction of the Minimum Distortion) 방식, menu vector 방식, OTS (One at a Time Search) 방식, 그리고 cross search 방식 등이 있다.[1-3] 이들은 탐색점 수를 줄여 계산량을 감축하는 방법인데 비하여 가산 투영을 이용한 블럭정합 알고리즘[5-7]은 탐색점의 수는 원래와 같고 블럭들의 유사도 검사시의 계산량을 줄이는 방법이다. 유사도 측정 계산량을 줄이기 위해 두단계의 정합 과정을 사용하였다. 첫번째 정합 단계에서는 블럭간의 2차원 정합 계산을 가산 투영을 이용하여 1차원 정합으로 변환하여 정합 계산량을 감축하였고 2차원 정합을 1차원 정합으로 변환할 때 발생하는 정보 손실로 인한 오정합을 보상하기 위해 두번째 정합 과정에서 2차원 정합을 사용하여 부정확한 정합으로 인한 화질 저하를 최소화하였다.[7]

FS BMA가 성능은 우수하나 방대한 양의 계산으로 실시간 처리에 많은 어려움이 따른다. 그러므로 이의 문제를 해결하기 위하여 앞에서 열거한 여러 가지 방식이 발표되었으나 이러한 고속 알고리즘의 개발에도 불구하고 이의 실시간 처리는 힘들기 때문에 최근에는 급속히 발전한 VLSI 기술을 이용하여 직접 FS BMA의 하드웨어 구조를 설계하여 실시간 처리가 가능하도록 하는 연구가 활발히 진행되고 있다.[8-14]

본 논문은 FS BMA 계산을 위해 기존의 구조보다 처리시간이 짧은 새로운 VLSI 구조 및 FS BMA보다 계산량이 적은 가산투영을 이용한 2단계 블럭정합 알고리즘 (이하 2단계 BMA)을 시스톨릭 어레이 (systolic array)를 이용하여 FS BMA보다 빠르게 구현한 VLSI 구조에 대하여 설명한다.

시스톨릭 어레이[15-16]는 VLSI기술을 이용하여 특정한 알고리즘의 수행 속도를 향상시키기 위해 최대한의 동시 실행 (concurrency)을 이룬 전용 하드웨어 구조로 이의 특징은 모듈성 (modularity), 규칙성 (regularity), 국부적 연결성 (local interconnection), 고도의 종속 연결성 (pipelining), 잘 동기된 다중처리 (multiprocessing) 등이다.

본 논문의 구성은 Ⅱ장에서 FS BMA와 2단계 BMA의 시스톨릭 어레이 구조를 제안하고, Ⅲ장에서 VHDL[17]을 이용한 모델링 및 시뮬레이션 결과를 보인다. 그리고 Ⅳ장에서 이의 성능을 분석하고, 마지막으로 Ⅴ장에서 결론을 맺는다.

Ⅱ. FS BMA와 2단계 BMA의 시스톨릭 어레이 구조

본 장에서는 FS BMA와 2단계 BMA의 시스톨릭 어레이 구조에 대하여 설명한다. 그리고 설명시 편의를 위하여 최대 변위 p를 2로 하고 부블럭의 크기는 3×3으로 하여 제안한 시스톨릭 어레이에 대하여 설명한다.

1. FS BMA

FS BMA는 계산량이 많음에도 불구하고 규칙적이기 때문에 VLSI 설계시 장점을 갖고 있다. 그러므로 시스톨릭 어레이를 이용한 VLSI 구조가 많이 발표되었다.[8-14]

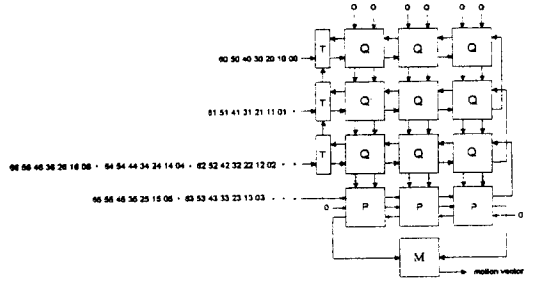


그림 1. FS BMA의 제안한 구조
Fig. 1. Proposed architecture for the FS BMA.

그림 1에 나타낸 시스톨릭 어레이는 [9]의 움직임 추정기인 AB2를 변형한 것으로 현재 영상의 화소값을 Q 블럭에 저장되어 있고 이전 영상의 화소값이 그림 1에 나타낸 바와 같이 Q 블럭으로 입력된다. Q 블럭에서는 입력된 화소값과 계산된 왜곡함수값을 출력한다. Q 블럭에서 계산된 왜곡함수값을 위에서 아래로 전파시켜 P 블럭에서 현재 영상과 이전 영상 사이의 왜곡함수값을 구해 M 블럭에서 최종 이동 벡터를 구한다. 제안한 구조는 [9]의 AB2 구조와 달리 이전 영상의 화소값이 좌측에서 우측으로 그리고 우측에서 좌측으로 두 방향으로 계산되므로 기존의 구조와 비슷한 PE (Processing Element)의 수로 기존의 구조보다 빠르게 동작한다.

그림 2는 FS BMA를 위해 제안한 구조에서 사용되는 PE의 기능을 나타낸 것으로 그림 2(a)는 Q 블럭을 나타내었다. Q 블럭은 현재영상의 화소값을 γ와 δ에 저장하고, 입력되는 이전영상의 화소값과 γ와 δ에 저장된 현재영상의 화소값 사이의 왜곡 함수를 계산하여 출력하고 입력값을 출력한다. 그림 2(b)는 T 블럭으로

한 클럭 지연시켜 입력값을 출력시킨다. 그림 2(c)는 P 블럭으로 T 블럭과 같이 입력을 한 클럭 지연시켜 출력한다. 그림 2(d)는 M 블럭으로 입력되는 두 왜곡함수값의 작은 값을 출력한다.

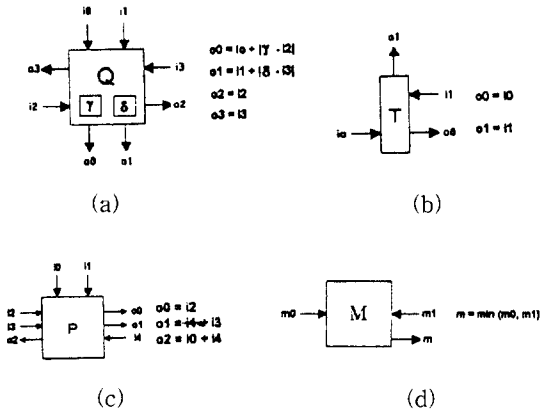


그림 2. FS BMA에 사용한 PE의 기능
(a) Q PE (b) T PE (c)P PE (d) M PE

Fig. 2. Function definition of the PE's employed in implementation of the FS BMA.
(a) Q PE, (b) T PE, (c) P PE, (d) M PE.

FS BMA에 필요한 클럭 수는 $(N + 2p)(p + 3)$이고 필요한 PE의 수는 Q PE가 N^2개, P와 T PE가 각각 N개, 그리고 한개의 M PE가 필요하므로 총 $N^2 + 2N + 1$개이다.

2. 2단계 BMA

2단계 BMA는 수평, 수직 가산투영값을 이용한 1차원 정합 과정과 여기서 얻어진 한개의 탐색점을 중심으로 주변 8개의 탐색점들과 중심점에 대해 2차원 왜곡함수값을 이용하여 정합점을 찾는 2차원 정합과정의 두 부분으로 구성하였다.[7] 그러므로 그림 3에 나타낸 것과 같이 제안한 구조도 1차원 정합 과정 블럭과 2차원 정합 과정 블럭으로 구성되어 있다.

H, V_AD, SR, 그리고 M 블럭으로 구성된 2단계 BMA의 1차원 정합 과정을 그림 4에 나타냈다. 그림 4에 점선으로 나타낸 이전 블럭의 화소값이 정해진 순서대로 H 블럭에 입력되어 수평 가산 투영값을 계산하고 V_AD 블럭에서 H 블럭으로부터 입력된 화소값을 이용하여 수직 가산 투영값을 구한다. 그리고 계산한 수평 및 수직 가산 투영값을 이용하여 최종적으로 V_AD 블럭에서 현재 영상과 이전 영상 사이의 1차원 왜곡 함수값을 구한다. 각 행의 V_AD 블럭에서 구한 1차원 왜곡 함수값들은 SR 블럭을 통하여 좌측에서 우측으로 이동하여 최종적으로 M 블럭에서 1차원 정합점을 구하게 된다.

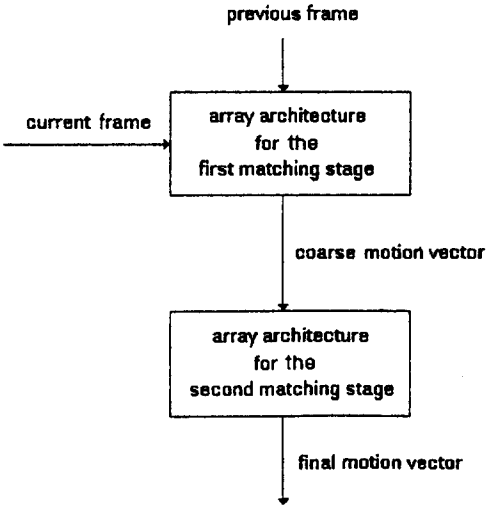


그림 3. 2단계 BMA의 제안한 구조
Fig. 3. Proposed architecture for the two-stage BMA.

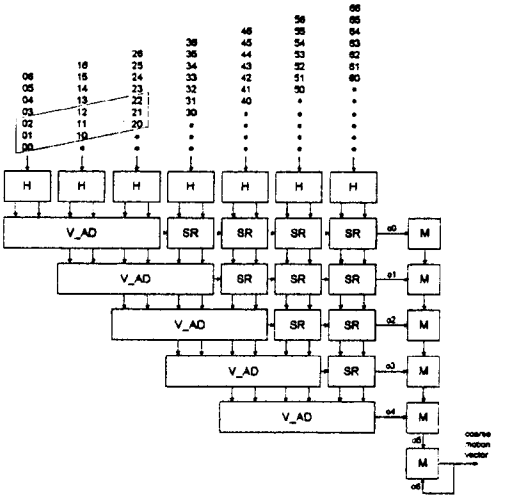


그림 4. 2단계 BMA의 1차원 정합 과정의 제안한 시스톨릭 어레이
Fig. 4. Proposed systolic array for the first matching stage of the two-stage BMA.

가능한 변위 p가 -2, -1, 0, +1, 그리고 +2이므로 그림 2의 o2가 수직 변위가 0이 된다. 그리고 수직 변위 -1 (+1)은 o1 (o3)에서 나타난다. 유사하게 수직 변위 -2 (+2)는 o0 (o4)에서 나타난다. 수평 변위는

마지막의 M 블럭이 세 번째 클럭에 바뀌게 되면 0이고 두번째 (네번째) 클럭에 바뀌면 수평 변위가 -1 (+1)이 된다. 유사하게 첫번째 (다섯번째) 클럭에 바뀌면 수평 변위가 -2 (+2)가 된다.

그림 5에 2단계 BMA에 사용된 PE를 나타내었다. 그림 5(a)~5(d)는 2단계 BMA의 1차원 정합과정에 사용된 PE로 그림 5(a)의 H 블럭은 입력되는 데이터를 α에 저장하여 수평 가산 투영값을 계산하고 입력 데이터를 출력 시켜 V_AD 블럭에서 수직 가산 투영값을 계산한다. 그림 5(b)의 V_AD 블럭은 H 블럭에서 전달된 데이터를 이용하여 수직 가산 투영값을 구하고 H 블럭에서 구한 수평 가산 투영값과 함께 1차원 왜곡함수값을 구한다. SR 블럭은 입력되는 i0, i1, 그리고 i2를 o0, o1, 그리고 o2로 전달한다. 그림 5(d)의 M 블럭은 두 입력중 작은 값을 출력한다. 그림 5(e)와 5(f)는 2단계 BMA의 2차원 정합 과정에 사용된 PE를 나타낸 것으로 그림 5(e)의 AD 블럭은 입력값인 i0과 i1의 왜곡 함수를 계산하여 β에 저장하고 i0과 i1을 o0과 o1로 출력한다. 그림 5(f)에 나타낸 N 블럭은 세 개의 입력중 가장 작은 값을 출력하여 최종 정합점을 구한다. 2차원 정합과정은 변위가 -1, 0, 그리고 +1 이므로 n1이 가장 작으면 수직 변위가 0이고 n0 (n2)가 가장 작으면 수직 변위가 -1 (+1)이 된다. 또한 N 블럭이 두 번째 클럭에 바뀌면 수평 변위가 0이고 첫번째 (세 번째) 클럭에서 변하면 수평 가산 투영이 +1 (-1)이 된다.

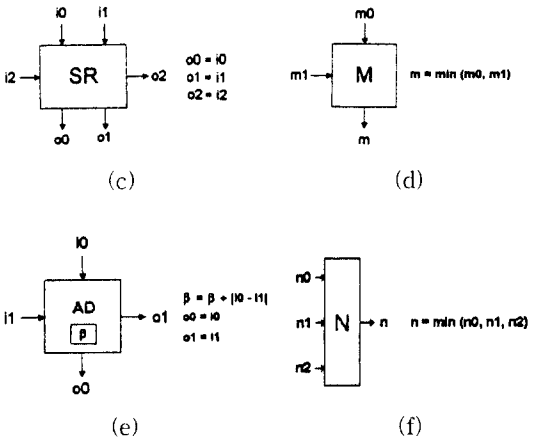


그림 5. 2단계 BMA에 사용한 PE의 기능
(a) H PE (b) V_AD PE (c) SR PE (d) M PE (e) AD PE (f) N PE

Fig. 5. Function definition of the PE's employed in implementation of the two-stage BMA.
(a) H PE, (b) V_AD PE, (c) SR PE, (d) M PE, (e) AD PE, (f) N PE.

2단계 BMA의 2차원 정합 과정은 기존의 구조인 [9]의 움직임 추정기와 [10]의 트리 구조를 사용하여 구할 수 있다. 그리고 2차원 정합과정을 위해 설계한 AD와 N 블럭으로 구성된 2단계 BMA의 2차원 정합 과정을 그림 6에 나타냈다. 2단계 BMA중 2차원 정합과정은 1차원 정합과정에서 구한 1차원 정합점과 그 주위의 8점에 대해 2차원 정합을 하는 것이다. 그러므로 2차원 정합과정은 FS BMA에서 최대 변위 p가 1인 것에 해당한다. 결과적으로 기존의 FS BMA 계산을 위한 VLSI 구조 [9-10]를 사용할 수 있고 본 논문의 그림 6과 같은 구조를 사용할 수 있다.

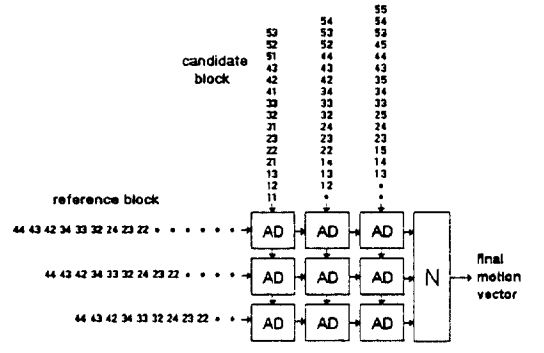


그림 6. 2단계 BMA의 2차원 정합 과정의 제안한 시스톨릭 어레이

Fig. 6. Proposed systolic array for the second matching stage of the two-stage BMA.

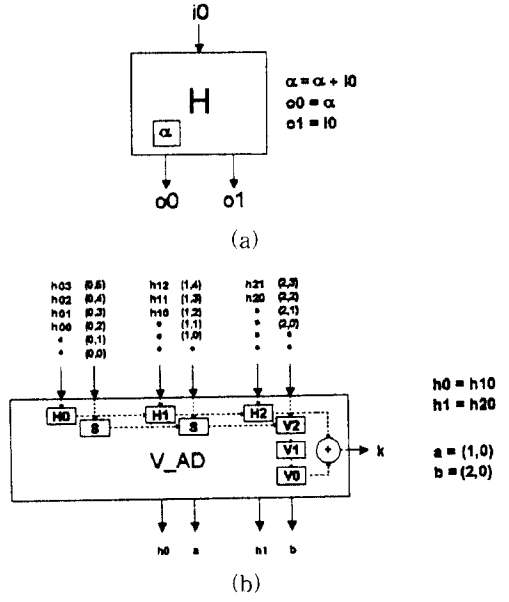


그림 6의 구조는 중심점과 주위의 8점을 전역 탐색할 수 있는 구조로 2차원 정합과정 및 부화소 탐색을 위하여 사용할 수 있다. 그림 6에서 현재 영상의 화소값은 좌측에서 우측으로 전파하고 이전 영상의 화소값은 위에서 아래로 전파하여 모든 계산을 수행하면 각각의 AD PE는 한 탐색점의 왜곡함수값을 저장한다. 모든 왜곡함수값이 계산되면 우측의 N PE로 전파하여 최종적으로 2차원 정합점을 구할 수 있다. 그러므로 그림 6의 구조는 최대 변위 p와 부 블럭 크기 N에 무관하게 항상 9개의 AD PE와 한개의 N PE로 구성된다.

결론적으로 2단계 BMA는 그림 4의 1차원 정합과정과 그림 6의 2차원 정합과정을 이용하여 구할 수 있다. 그리고 제안한 구조의 설명의 편의를 위하여 각각의 구조가 sequential하게 작동하는 것으로 가정하였지만 실제의 동작은 그림 4의 1차원 정합과정이 한 블럭에 대해 계산하고 있으면 그림 6의 2차원 정합과정은 그 전 블럭의 2차원 정합과정을 수행한다.

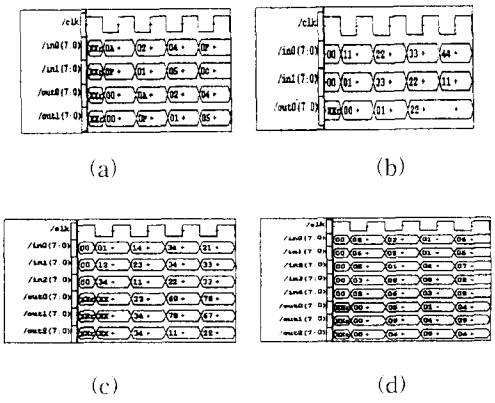
2단계 BMA 계산에 소요되는 클럭은 1차원 정합 과정에 $2(N+1)+6p$이고 2차원 정합 과정에 N^2+N-1이 필요하므로 전체적으로 $N(N+3)+6p+1$이다. 또한 필요한 PE의 수는 1차원 정합 과정에 H PE가 $2p+N$개, V_AD PE가 $2p+1$개, SR PE가 $p(2p+1)$개, 그리고 M PE가 $2p+2$개 필요하므로 $N+2p^2+7p+3$개이고 2차원 정합 과정에 9개의 AD PE와 1개의 N PE가 필요하므로 총 $N+2p^2+7p+13$개이다.

III. 제안한 구조의 VHDL 시뮬레이션

본 장에서는 제안한 구조를 VHDL를 이용하여 모델링하고 시뮬레이션 결과에 관하여 설명한다

그림 7은 FS BMA에서 사용한 T, M, Q, 그리고 P PE의 시뮬레이션 결과로 그림 2에 정의한 대로 올바르게 동작함을 확인할 수 있다. 예로, 그림 7(d)의 P PE는 /in2를 /out0에 전달하고 /out1 (/out2)는 /in1과 /in3 (/in0과 /in4)의 합이 한 클럭 후에 출력된다.

그림 8은 부 블럭의 크기 N은 4이고 최대 변위 p는 3인 경우의 FS BMA의 시뮬레이션 결과로 /rst, /clk, /row, 그리고 /col은 각각 reset 신호, 클럭 신호, 행과 열의 위치를 나타내고 M 블럭에 입력되는 현재 영상과 이전 영상 사이의 왜곡 함수값은 /oi, $0 < i < 1$.에 나타냈다. 그리고 현재 영상의 위치 값은 (4, 4)이고 계산 결과의 움직임 벡터는 /mvx와 /mvy에 (3, 1)로 나타나 있다.



(a) (b) (c) (d)

그림 7. FS BMA에 사용한 기본 PE의 VHDL 시뮬레이션 결과
(a) T PE (b) M PE (c) Q PE (d) P PE

Fig. 7. VHDL simulation results of the basic PE's for the FS BMA.
(a) T PE, (b) M PE, (c) Q PE, (d) P PE.

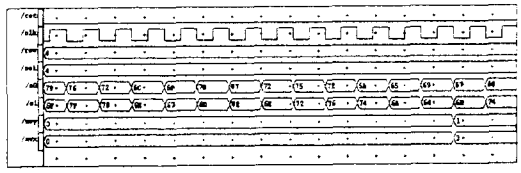


그림 8. FS BMA의 VHDL 시뮬레이션 결과

Fig. 8. VHDL simulation result of the FS BMA.

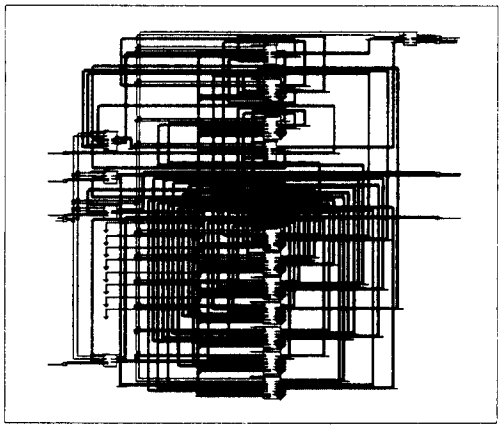
그림 9는 FS BMA와 2단계 BMA를 VGC450 라이브러리를 이용하여 최적화한 결과로 그림 9(a)는 FS BMA를 나타낸 것으로 Q, P, T, 그리고 M 블럭으로 구성되어 있다. 그림 9(b)는 1차원 정합 과정을 나타내는 것으로 H, V_AD, SR, 그리고 M 블럭으로 구성되어 있으며 그림 9(c)는 2차원 정합 과정의 실험 결과로 AD와 N PE로 구성되어 있다.

표 1에 제안한 구조에 필요한 게이트 수를 최적화한 결과를 이용하여 추정한 결과를 나타내었다. FS BMA에 필요한 Q, T, P, 그리고 M PE에 각각 886, 210, 297, 그리고 261개가 필요하다. 그리고 2단계 BMA의 1차원 정합 과정에 필요한 H, V_AD, SR, 그리고 M PE에 각각 448, 1969, 198, 그리고 187개가 필요하고 2차원 정합 과정에 필요한 AD와 N PE는 각각 371개와 5260개가 필요하다. 그러므로 $N=16$이고 $p=7$일 때 필요한 게이트 수는 FS BMA에 235190개가

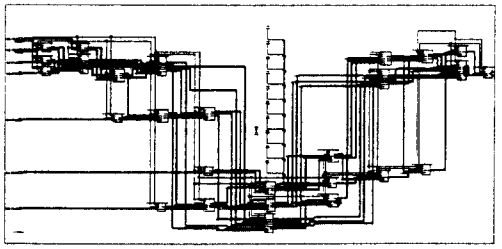
필요하다. 그리고 2단계 BMA의 1차원 정합 과정에 66757개 필요하고 2차원 정합 과정에 8599개가 필요하므로 전체적으로는 75356개가 필요하다.

표 1. 필요한 게이트의 수 (N=16, p=7)
Table 1. Required number of gates (N=16, p=7).

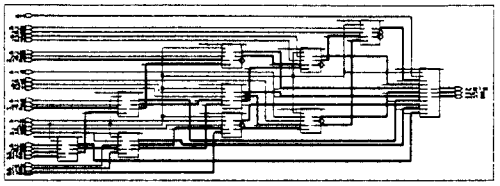
방 법		PE	게이트의 수
FS BMA		Q	886
		T	210
		P	297
		M	261
전체 (추정)			235190
2단계 BMA	1차원 정합과정	H	448
		V_AD	1969
		SR	198
		M	187
		전체 (추정)	66757
	2차원 정합과정	AD	371
		N	5260
		전체 (추정)	8599
전체 (추정)			75356



(a)



(b)



(c)

그림 9. VHDL 최적화 결과
(a) FS BMA (b) 2단계 BMA의 1단계 정합과정 (c) 2단계 BMA의 2단계 정합과정
Fig. 9. VHDL optimization results.
(a) FS BMA,(b) First matching stage of the two-stage BMA, (c) Second matching stage of the two-stage BMA.

Ⅳ. 성능 분석

제안한 구조의 성능을 평가하기 위하여 기존의 구조인 [9]의 움직임 추정기인 AB1, AB2, 그리고 AS2 구조와 [10]의 완전 트리와 16-cut 트리의 필요한 클럭수와 PE 수를 표 2와 표 3에 나타내었다.

표 2. 필요한 클럭 수
Table 2. Required number of clocks.

방법			클럭 수	(N, p)			
				(8,7)	(8,15)	(16,7)	(16,15)
FS BMA		AB1	$N(2p+1)(N+2p)$	2640	9424	7200	22816
		AB2	$(2p+1)(N+2p)$	330	1178	450	1426
		AS2	$N(N+2p)$	176	304	480	736
		full tree	$(2p+1)^2+2\log_2 N+1$	232	968	234	970
		16-cut tree	$16(2p+1)^2+2\log_2 N+1)$	3712	15488	3744	15520
		제안한 방법	$(N+2p)(p+3)$	220	684	300	828
2단계 BMA	1차원 정합 과정		$2N+6p+2$	60	108	76	124
	2차원 정합 과정	AB1	$3N(N+2)$	240	240	864	864
		AB2	$3(N+2)$	30	30	54	54
		AS2	$N(N+2)$	80	80	288	288
		full tree	$2\log_2 N+10$	16	16	18	18
		16-cut tree	$16(2\log_2 N+10)$	256	256	288	288
		제안한 방법	N^2+N+1	71	71	271	271

표 2는 기존의 구조와 제안한 구조를 N과 p의 함수로 필요한 클럭수를 나타낸 것으로 한 클럭은 하나의 PE에서 소요되는 시간을 의미한다. 완전 트리 구조는

소요되는 클럭수는 적지만 하드웨어 비용이 너무 커 16-cut 트리 구조도 표 2에 나타내었다. 2단계 BMA의 1차원 정합 과정은 그림 2의 구조를 나타낸 것이고 2차원 정합 과정은 [9]의 AB1, AB2, AS2 구조, [10]의 완전 트리, 16-cut 트리, 그리고 그림 6의 제안한 구조를 나타내었다. 또한 기존 움직임 추정기에 널리 사용되는 (N, p) = (8, 7), (8, 15), (16, 7), 그리고 (16, 15)인 경우도 표 2에 추가하였다. 표 3은 필요한 PE의 수를 나타낸 것으로 표 2에서는 나쁜 성능을 나타내는 16-cut 트리와 AB1 구조가 가장 적음을 알 수 있다.

표 3. 필요한 PE 수
Table 3. Required number of PE's.

방 법			PE의 수	(N, p)			
				(8,7)	(8,15)	(16,7)	(16, 15)
FS BMA		AB1	$N+2$	10	10	18	18
		AB2	N^2+N+1	73	73	273	273
		AS2	$(N+2)(2p+1)+1$	151	311	271	559
		full tree	$2N^2$	128	128	512	512
		16-cut tree	$N^2/8$	8	8	32	32
		제안한 방법	$N^2 + 2N + 1$	81	81	289	289
2단계 BMA	1차원 정합 과정		$N+2p^2+7p+3$	158	566	166	574
	2차원 정합 과정	AB1	$N+2$	10	10	18	18
		AB2	N^2+N+1	73	73	273	273
		AS2	$3(N+2)+1$	31	31	55	55
		full tree	$2N^2$	128	128	512	512
		16 cut tree	$N^2/8$	8	8	32	32
		제안한 방법	10	10	10	10	10

FS BMA를 위해 제안한 구조와 기존의 구조를 비교하면 제안한 구조가 클럭 수에서는 완전 트리 구조와 함께 가장 적게 필요함을 알 수 있고 필요한 PE의 수에서는 처리 속도에서 나쁜 성능을 나타내는 AB1 구조와 16-cut 트리 구조가 좋은 성능을 나타낸다. 그리고 필요한 클럭 수에서 제안한 구조와 비슷한 성능을 나타내는 완전 트리 구조는 제안한 구조보다 필요한 PE의 수는 많음을 알 수 있다.

2단계 BMA의 경우에는 1차원 정합 과정은 제안한 구조를 사용하고 2차원 정합 과정은 완전 트리와 AB2 또는 AS2 구조를 사용하는 경우가 2차원 정합 과정을 제안한 구조를 사용하는 경우보다 빠르게 동작하지만 표 3처럼 제안한 구조가 10개인데 반하여 AB2, AS2 구조와 완전 트리 구조는 31개에서 512개가 필요함을 알 수 있다.

표 3에 나타낸 구조인 AB1, AB2, AS2, 완전 트리, 16-cut 트리, 그리고 제안한 구조의 (N, p) = (16, 7)일 때 필요한 덧셈기 수는 각각 34, 529, 511, 512, 32, 545이다. 그리고 1차원 정합 과정에 필요한 덧셈기 수는 286이고 2차원 정합 과정을 AB1, AB2, AS2, 완전 트리, 16-cut 트리, 그리고 제안한 구조에 필요한 덧셈기 수는 각각 34, 529, 103, 512, 32, 20이다.

V. 결 론

본 논문에서는 FS BMA와 FS BMA에 비하여 성능이 비슷하고 계산량을 현저하게 줄인 2단계 BMA의 시스톨릭 어레이 구조를 제안하였다. 제안한 구조는 단순한 하드웨어 구조로 기존의 구조보다 빠르게 동작한다. 또한 VHDL를 이용하여 모델링하고 시뮬레이션하여 제안한 구조가 정상적으로 동작함을 확인하였다. 그리고 VHDL 최적화 과정을 거쳐 제안한 구조에 필요한 게이트 수를 구하였다. 앞으로는 FS BMA와 2단계 BMA를 위해 제안한 시스톨릭 어레이 구조를 비슷한 하드웨어 비용으로 좀 더 빠르게 동작하도록 하는 연구가 진행되어야 한다. 그리고 제안한 구조의 데이타 bandwidth를 줄이는 연구가 진행되어야 하고 제안한 구조를 실제적인 칩으로 제작하는 연구를 진행하여야 하겠다.

참 고 문 헌

[1] H. G. Musmann, P. Pirsch, and H.-J. Grallert, "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] T. Koga *et al.*, "Motion-compensated interframe coding for video conferencing," in

Proc. Nat. Telecom. Conf., pp. G 5.3.1-G 5.3.5, Nov./Dec. 1981.

[5] J.-S. Kim and R.-H. Park, "Featured based block matching algorithm using integral projections" *IEE Electron. Lett.*, vol. 25, pp. 214-215, Feb. 1989.

[6] 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.

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

[8] P. Pirsch, N. Demassieux, and W. Gehrke, "VLSI architectures for video compression -A survey," *Proc. IEEE*, vol. 83, pp. 220-246, Feb. 1995.

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

[10] 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.

[11] 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.

[12] L. De 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. CSVT-2, pp. 169-175, June 1992.

[14] S. B. Pan, S. S. Chae, and R.-H. Park, "VLSI architectures for block matching algorithms using systolic arrays," *IEEE Trans. Circuits Syst. for Video Tech.*, vol. CSVT-6, pp. 67-73, Feb. 1996.

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

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

[17] *IEEE Standard VHDL Language Reference Manual*. New York: IEEE Inc., 1994.

저 자 소 개

潘 璧 範(準會員) 第 31卷 B編 第 7號 參照
현재 서강대학교 전자공학과 대학원 박사과정 재학중

蔡 承 秀(準會員) 第 31卷 B編 第 7號 參照
현재 삼성전자 근무

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