論文 96-21-2-2

선형 시스토릭 어레이를 이용한 완전탐색 블럭정합 이동 예측기의 구조

正會員 金 起 賢*, 李 起 徹**

A Linear Systolic Array Based Architecture for Full-search Block Matching Motion Estimator

Ki-Hyun Kim*, Kichul Kim** *Regular Members*

要 約

본 논문에서는 선형 시스토릭 어레이(Linear Systolic Array)에 기초한 완전탐색 블럭정합 이동 예측기(Full-search Block Matching Motion Estimator)의 새로운 구조를 제안한다. 제안하는 이동 예측기에 사용된 선형 시스토릭 어레이는 입력 데이터뿐만 아니라 제어 신호까지 레지스터를 통한 파이프라인 형태로 내부 프로세서에 공급함으로써 버스의 사용을 제한하여 고속의 동작이 가능하다. 또한 외부 입력 데이터가 프로세서 동작 속도의 절반의 속도로 입력되게 함으로써 입력 데이터 대역폭의 요구량이 낮다. 제안한 이동 예측기는 블럭 크기의 변화와 탐색범위의 증가에 대해 요구되는 프로세서의 수와 입력 대역폭 면에서 기존의 이동 예측기들보다 우수한 확장성을 가지고 있다.

ABSTRACT

This paper presents a new architecture for full-search block-matching motion estimation. The architecture is based on linear systolic arrays. High speed operation is obtained by feeding reference data, search data, and control signals into the linear systolic array in a pipelined fashion. Input data are fed into the linear systolic array at a half of the processor speed, reducing the required data bandwidth to half. The proposed architecture has a good scalability with respect to the number of processors and input bandwidth when the size of reference block and search range change.

*한국전자통신연구소 컴퓨터연구단 미디어연구실, 선임연구원

**서울시립대학교 반도체공학과 전임강사

論文番號:95323-0916

接受日字:1995년 9월 16일

Ⅰ. 서 론

최근 들어 디지털 비디오 신호를 이용한 화상회의, 화상전화, HDTV와 같은 동영상의 이용에 대한 연구

개발이 활발하게 진행되고 있다. 동영상은 매우 많은 양의 정보를 가지고 있기 때문에 동영상을 그대로 전송하거나 저장하려면 고속의 전송 선로와 매우 큰 용량의 저장 매체가 필요하게 된다. 따라서 동영상의 효율적인 전송과 저장을 위해서는 데이터의 압축은 필수적이다[1]~[4].

동영상의 압축에는 동영상 데이터에 존재하는 중복성(redundancy)이 이용된다. 동영상 데이터의 중복성에는 한 영상내(intraframe)에 존재하는 공간상의 중복성과 연속된 영상들 사이(interframe)에 존재하는 시간축상의 중복성이 있다. 공간상의 중복성을 이용한 데이터 압축방법으로는 DCT(Discret Cosine Transform) 방법이 널리 이용되며, 시간축상의 중복성을 이용한 압축방법으로는 이동 예측기(Motion Estimator)를 이용한 이동 보상(Motion Compensation) 방법이 널리 이용되고 있다.

이동 예측 방법에는 전체 화면을 몇개의 블럭으로 나누어 블럭 단위로 이동벡터를 찾아내는 블럭정합 알고리듬(BMA : Block Matching Algorithm)이 가장 널리 알려져 있다[5]~[8]. 블럭정합 알고리듬을 구현하는 방법에는 탐색영역의 모든 후보 블럭을 대상으로 블럭정합을 수행하는 완전탐색 블럭정합 알고리듬(FBMA : Full-search Block Matching Algorithm)[9][10], 모든 후보 블럭을 대상으로 탐색을 수행하되 블럭 내의 화소 중 일부만 정합에 사용하는 부분 오류 추출 전영역 탐색 알고리듬[7], 대략적인 이동 벡터에 해당되는 후보블럭들에 대한 블럭정합으로부터 출발하여 단계적으로 정확한 이동 벡터를 찾아가는 계층적 탐색(Hierachical Search)[8] 등이 있다.

상기한 알고리듬들 중에서 탐색에 필요한 계산량에서 계층적 탐색이 일반적으로 유리하고, 완전탐색 블럭정합 알고리듬은 정확도에서 가장 우수하며 대이터의 흐름이 용이하여 널리 사용되고 있다[7].

그림 1은 완전탐색 블럭정합 알고리듬을 나타내고 있다. 그림 1에서 현재 화면은 크기가 NxN인 기준블럭 단위로 나누어진다. 나누어진 기준블럭 행렬에서 한 기준블럭 행(row)을 슬라이스(slice)라 한다. 이전 화면에서 기준블럭과 같은 위치에서 수평 범위는 -p~p-1, 수직 범위는 -q~q-1의 탐색 범위내의 영역을 탐색영역이라 한다. 탐색영역내의 후보블럭들중 가장 작은 MAD(Mean Absolute Difference)값을 갖는 블럭의 이동벡터가 기준블럭의 이동벡터가 된다.

그림 1의 완전탐색 블럭정합 알고리듬을 수식으로 표현하면 식 (1)과 같다.

$$MAD(k,l)=\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}|r(i,j)-s(k+i,l+j)| \qquad (1)$$

$$MV=(k,l)_{\min(MAD(k,l))}$$

단, $-p\le k<p,\ -q\le l<q$.

식 (1)에서 r(i, j)는 기준블럭 데이터이며, s(k + i, l + j)는 한 후보블럭의 데이터이다. 한 기준블럭의 이동 벡터 MV는 탐색영역내의 가능한 모든 후보블럭의 이동 벡터 (k, l) 중에서 가장 작은 MAD 값을 갖는 벡터가 된다.

완전탐색 블럭정합 알고리듬은 정확도가 높은 반면 식 (1)에서와 같이 계산량이 많기 때문에 하나의 연산기로서는 실시간 처리가 불가능하여 여러개의 연산기를 사용하는 병렬처리가 필요하다. 또한 기준블럭의 크기가 NxN일 경우, 한 이동 벡터의 MAD 값 구하기 위해서 $2N^2$의 데이터가 필요하므로 인접한 이동 벡터간 또는 인접한 기준블럭간에 데이터를 공유하여 입력 데이터의 대역폭을 줄일 수 있는 효율적인 병렬 구조가 요구된다. 따라서 VLSI의 구현에 적합하면서 실시간 처리가 가능한 병렬구조에 많은 연구가 이루어지고 있다[9]~[13].

최근에는 시스톨릭 어레이(systolic array) 또는 세

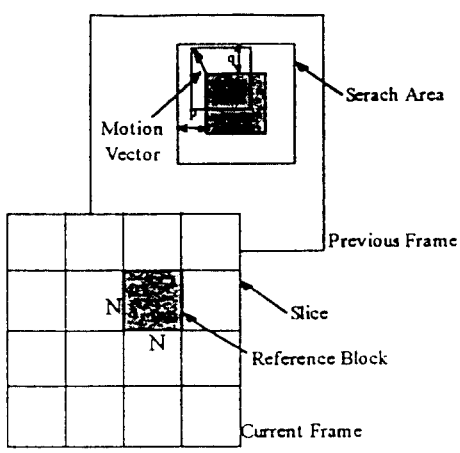


그림 1. 완전탐색 블럭정합 알고리즘
Fig. 1. Full-search Block Matching Algorithm

미시스토릭(semisystolic array) 어레이를 이용한 병렬 구조로 이동 예측기를 구현하는 방법이 활발하게 연구되고 있다[9]~[13]. Komarek은 시스토릭 어레이를 사용한 이동 예측기의 구조를 제안하였다[10]. Komarek이 제안한 1차원과 2차원 어레이에서는 사용되는 모든 데이터가 데이터 버스가아닌 파이프라인 형태로 연산기에 공급됨으로써 제어가 간단하며 고속 동작이 가능하다. 그러나 시스토릭 어레이의 고속 동작을 유지하는데 필요한 대량의 데이터를 어레이에 입력시키기 위하여 입력 핀의 수가 많아지는 단점이 있다. Yang은 1차원 세미시스토릭 어레이를 사용한 이동 예측기의 구조를 제안하였다[4, 11]. Yang이 제안한 1차원 어레이는 독립적인 한 모듈로 설계되어 기준블럭 크기 변화에 따라 시스템 구성이 용이하며, 순차적인 데이터 입력 방법으로 입력 핀의 수가 작은 장점이 있다. 그러나 입력된 데이터의 활용도가 낮아 HDTV와 같은 고속처리를 위한 응용에는 적합하지 않다. Hsieh는 2차원 시스토릭 어레이와 쉬프터 레지스터(Shift Register)를 사용한 이동 예측기의 구조를 제안하였다[10]. Hsieh가 제안한 구조는 쉬프터 레지스터를 이용하여 시스토릭 어레이에 데이터를 공급함으로써 순차적인 데이터 입력을 가능하게 하여 입력 핀의 수를 줄였다. 그러나 입력된 데이터의 활용도가 낮아 고속 처리를 위한 응용에는 적합하지 않다.

본 논문에서는 선형 시스토릭 어레이를 이용한 이동 예측기의 새로운 구조를 제안한다. 제안되는 구조는 상기에서 기술한 각 알고리듬의 장점(Komarek 알고리듬의 파이프라인으로 데이터를 공급, Yang의 알고리듬의 독립적인 모듈 구조, Hsieh 알고리듬의 쉬프터 레지스터를 이용한 순차적 데이터 공급)등을 이용하여 각 알고리즘의 단점을 보완한 구조이다. 제안되는 구조는 시스토릭 어레이의 장점인 규칙성, 고속 동작 특성과 함께 VLSI로 구현하기에 적당한 다음과 같은 장점을 가지고 있다.

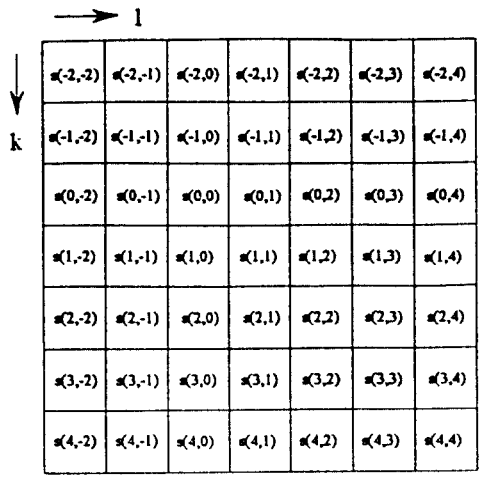
- 데이터가 프로세서 동작속도의 절반의 속도로 입력됨으로써 입력 데이터의 대역폭에 대한 요구량이 낮다.
- 기준블럭 데이터와 탐색영역 데이터가 열단위로 한번씩만 입력되면 모든 기준블럭에 대해서 탐색영역 범위의 모든 이동 벡터에 대한 MAD 연산이 수행되므로 데이터의 활용도(utilization)가 높으며, 데이터의 공급과 제어가 쉽다.
- 전체적인 시스템 구성이 간단하여 탐색범위가 변화할 때 프로세서의 수 및 입력핀 수의 면에서 확장성이 우수하다.

본 논문의 나머지 부분은 다음과 같이 구성되어 있다. 2장에서는 낮은 입력 대역폭을 요구하는 선형 시스토릭 어레이에 대해서 설명한다. 3장에서는 선형 시스토릭 어레이를 기반으로 하며, 수평방향 탐색범위의 변화에 대해 적응성이 우수한 이동 예측기의 구조를 제안한다. 4장에서는 제안된 이동 예측기가 수직방향 탐색범위의 변화에 대하여 우수한 확장성을 가지고 있음을 설명한다. 5장에서는 제안된 이동 예측기가 가지는 성능에 대해 설명한다. 끝으로 6장에서 본 논문의 결론을 보인다.

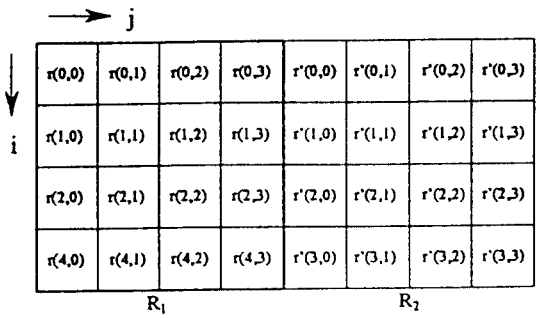
Ⅱ. 낮은 입력 데이터 대역폭을 요구하는 선형 시스토릭 어레이

본 장에서는 입력된 데이터를 반복하여 이용함으로써 프로세서가 동작하는 반의 속도로 데이터 입력이 가능한 낮은 입력 데이터 대역폭에서 동작하는 선형 시스토릭 어레이에 대해서 설명한다. 본 장에서는 기준블럭의 크기가 4x4이고, 탐색범위가 -2~1 인경우를 예로 들어 선형 시스토릭 어레이의 동작을 상세히 보인다. 그림 2 (a)에 탐색영역이 나타나 있으며, 그림 2 (b)에 2개의 이웃한 기준블럭 R_1과 R_2가 나타나 있다. 그림 2(a)에서 s(-2, -2)는 이전 화면에서 기준블럭 R1에 대해 (-2, -2)의 이동 벡터를 가지는 후보블럭의 좌측상단에 있는 화소(pixel)이며 s(-2, 2)는 이전 화면에서 기준블럭 R_2에 대해 (-2, -2)의 이동 벡터를 가지는 후보블럭의 좌측상단에 있는 화소(pixel)이다. 그림 2(a)에서 s(-2, 2) 이후의 탐색영역 데이터는 기준블럭 R_1과 R_2의 MAD 연산에 공통으로 사용된다.

그림 3에 본 논문에서 제안하는 선형 시스토릭 어레이와 선형 시스토릭 어레이에 사용되는 프로세서(PE:Processing Element)가 나타나 있다. 그림 3 (a)에서 R은 레지스터이고, M은 멀티프렉서(multiplexor)이며, AD(absolute difference)는 입력된 두 값 차이의 절대값을 계산하는 부분이다. 각 프로세서에는 기준블럭 데이터를 왼쪽 프로세서(PE_{i-1})로부터 받아 오른쪽 프로세서(PE_{i+1})로 전달하는 레지스터가 하나



(a) 7 x 7 탐색영역 데이터.



(b) 4 x 4 기준블럭 데이터.

그림 2. 탐색영역과 기준블럭 데이터의 예.
Fig. 2. An example of Search area and Reference data.

있으며 탐색영역 데이터를 오른쪽 프로세서(PE_{i+1})로부터 받아 왼쪽 프로세서(PE_{i-1})로 전달하는 레지스터가 2개 있다. 또한 연산중인 MAD의 값을 저장하는 두 개의 레지스터와 계산 완료된 MAD값을 왼쪽 프로세서(PE_{i-1})로 전달하는 레지스터가 있다. 각 프로세서에는 제어신호를 저장, 전달하는 레지스터가 있으며, 모든 제어신호는 왼쪽 프로세서(PE_{i-1})에서 입력되어 오른쪽 프로세서(PE_{i+1})로 전달된다. 각 프로세서는 두 개의 이동벡터를 위한 두 개의 MAD 값을 구하는 연산을 동시에 수행한다. 즉, 프로세서 PE_i가 짝수 클럭에 하나의 MAD를 위한 연산을 수행한다면, 홀 수 클럭에는 다른 하나의 MAD를 위한 연산을 수행한다.

그림 3 (b)의 선형 시스토릭 어레이에는 MAD 연산을 위한 세개의 데이터 경로가 파이프라인 상태(pipelined mode)로 데이터를 공급한다. 세개의 데이터 경로중 한개는 기준 블럭의 데이터를 공급하기 위한 것이며 두개는 탐색영역의 데이타을 위한 것이다.

두개의 탐색영역 데이터를 위한 경로를 각각 path 0와 path 1으로 칭한다. 그림 3 (b)에는 제어신호를 위한 경로와 MAD값을 전달하기위한 경로가 생략되어 있다. 탐색영역 데이터는 열(column)단위로 선형 어레이의 오른쪽에서 왼쪽으로 전달되며, 기준블럭 데이터는 현재의 열(column)과 지연 버퍼(delay buffer)를 통한 이전의 열이 멀티플렉서를 통하여 교대로 선형 어레이의 왼쪽에서 오른쪽으로 전달된다.

본 장의 나머지 부분에서는 제안된 선형 시스토릭 어레이의 동작에 대해서 그림 2의 예를 이용하여 설명한다. 본 예에서 선형 시스토릭 어레이의 프로세서 PE3는 모든 기준블럭에 대하여 두개의 이동 벡터 (i-2, -2), (i-2, -1)에 대한 MAD의 값들을 계산한다. 위에서 설명한 바와 같이 각 프로세서에는 MAD 값을 누적하는 레지스터가 두개씩 있으며 두개의 이동 벡터에 대한 MAD의 부분합이 매 클럭마다 교대로 계산되어 누적된다. 선형 시스토릭 어레이에 탐색영역 데이터를 공급하는 방법은 아래와 같다.

1. 하나의 탐색영역 데이터 s(l, k)는 연속된 2 클럭 동안 PE_1에 공급되며, PE_i는 매 클럭마다 PE_{i+1}에서 탐색영역 데이터를 공급받는다.
2. 탐색영역의 짝수열(even cloumn) 데이터(s(k, -2), s(k, 0),..)는 PE_3의 path 0을 통해서 입력되고, 홀수열(odd column) 데이터(s(k, -1), s(k, 1)...)는 PE3의 path 1을 통해서 입력된다.
3. 짝수열 탐색영역 데이터가 PE_3의 path 0에 공급된지 8 클럭 뒤에 이어지는 홀수 열 데이터가 PE_3의 path 1에 공급된다.

기준블럭 데이터는 아래의 규칙에 따라 선형 시스토릭 어레이에 공급된다.

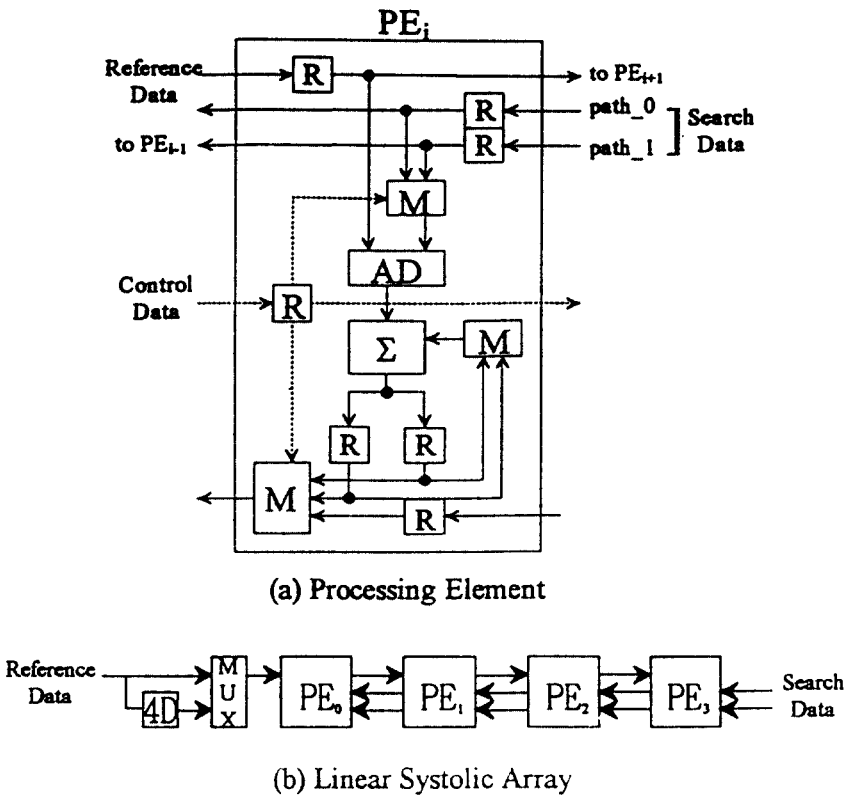


그림 3. 프로세서 구조와 선형 시스토릭 어레이
Fig. 3. Processing Element and Linear Systolic Array

1. 기준블럭 데이터 r(i, j)와 8 클럭의 지연을 통해서 나타나는 r(i, j-1)은 멀티플렉서를 통하여 교대로 PE_0에 공급된다. 단 j=0 인 경우에는 하나의 열만이 공급되거나 이전 기준블럭의 열과 현재 기준블럭의 열이 교대로 공급된다.
2. PE_{i+1}은 매클럭마다 PE_i에서 기준블럭 데이터를 공급받는다.
3. 기준블럭 데이터 r(0, j)는 탐색영역 데이터 s(-2, k)가 PE_3의 path_0에 공급된지 4 클럭만에 PE_0에 공급된다.

그림 4에 선형 시스토릭 어레이로 공급되는 데이터와 각 프로세서에서 이루어지는 연산의 예가 나타나 있다. 그림 4의 상단에는 각 프로세서에서 수행되어야 하는 연산이 수식으로 나타나 있으며, 각 프로세서의 열에는 매 클럭마다 프로세서에 공급되는 데이터가 나타나 있다.

Path_0 열에는 각 프로세서에 path_0를 통하여 공급되는 데이터가 나타나 있으며, Path_1 열에는 path_1를 통하여 공급되는 데이터가 나타나 있다. 회색으로 표시된 부분은 MAD값을 계산하기위한 유효한 연산이 이루어지는 부분을 나타내고 있으며 이때 사용되는 기준블럭의 데이터가 탐색영역 데이터와 함께 나타나 있다. MAD(i-2, -2), $0 \le i < 4$ 값을 위한 연산에 사용되는 데이터는 로마체로 나타나 있으며, MAD(i-2, -1), $0 \le i < 4$ 값을 위한 연산에 사용되는 데이터는 이텔릭체로 표시되어 있다. 두 개의 MAD를 위한 부분합은 서로 다른 레지스터에 저장되며, 연산이 완료된 MAD 값은 MAD 전달 레지스터를 통하여 왼쪽 프로세서(PE_{i-1})로 전달된다.

Cycle time	PE0 ΣΣ|r(i,j)-s(-2+i,-2+j)| ΣΣ|r(i,j)-s(-2+i,-1+j)|		PE1 ΣΣ|r(i,j)-s(-1+i,-2+j)| ΣΣ|r(i,j)-s(-1+i,-1+j)|		PE2 ΣΣ|r(i,j)-s(0+i,-2+j)| ΣΣ|r(i,j)-s(-0+i,-1+j)|		PE3 ΣΣ|r(i,j)-s(1+i,-2+j)| ΣΣ|r(i,j)-s(1+i,-1+j)|	
t	Path 0	Path 1	Path 0	Path 1	Path 0	Path 1	Path 0	Path 1
0							s(-2,-2)	
1					s(-2,-2)		s(-2,-2)	
2			s(-2,-2)		s(-2,-2)		s(-1,-2)	
3+0x8+0	r(0,0)-s(-2,-2)		s(-2,-2)		s(-1,-2)		s(-1,-2)	
3+0x8+1	s(-2,-2)		r(0,0)-s(-1,-2)		s(-1,-2)		s(0,-2)	
3+0x8+2	r(1,0)-s(-1,-2)		s(-1,-2)		r(0,0)-s(0,-2)		s(0,-2)	
3+0x8+3	s(-1,-2)		r(1,0)-s(0,-2)		s(0,-2)		r(0,0)-s(1,-2)	
3+0x8+4	r(2,0)-s(0,-2)		s(0,-2)		r(1,0)-s(1,-2)		s(1,-2)	
3+0x8+5	s(0,-2)		r(2,0)-s(1,-2)		s(1,-2)		r(1,0)-s(2,-2)	s(-2,-1)
3+0x8+6	r(3,0)-s(1,-2)		s(1,-2)		r(2,0)-s(2,-2)	s(-2,-1)	s(2,-2)	s(-2,-1)
3+0x8+7	s(1,-2)		r(3,0)-s(2,-2)	s(-2,-1)	s(2,-2)	s(-2,-1)	r(2,0)-s(3,-2)	s(-1,-1)
3+1x8+0	s(2,-2)	r(0,1)-s(-2,-1)	s(2,-2)	s(-2,-1)	r(3,0)-s(3,-2)	s(-1,-1)	s(3,-2)	s(-1,-1)
3+1x8+1	s(2,-2)	r(0,0)-s(-2,-1)	s(3,-2)	r(0,1)-s(-1,-1)	s(3,-2)	s(-1,-1)	r(3,0)-s(4,-2)	s(0,-1)
3+1x8+2	s(3,-2)	r(1,1)-s(-1,-1)	s(3,-2)	r(0,0)-s(-1,-1)	s(4,-2)	r(0,1)-s(0,-1)	s(4,-2)	s(0,-1)
3+1x8+3	s(3,-2)	r(1,0)-s(-1,-1)	s(4,-2)	r(1,1)-s(0,-1)	s(4,-2)	r(0,0)-s(0,-1)		r(0,1)-s(1,-1)
3+1x8+4	s(4,-2)	r(2,1)-s(0,-1)	s(4,-2)	r(1,0)-s(0,-1)		r(1,1)-s(1,-1)		r(0,0)-s(1,-1)
3+1x8+5	s(4,-2)	r(2,0)-s(0,-1)		r(2,1)-s(1,-1)		r(1,0)-s(1,-1)	s(-2,0)	r(1,1)-s(2,-1)
3+1x8+6		r(3,1)-s(1,-1)		r(2,0)-s(1,-1)	s(-2,0)	r(2,1)-s(2,-1)	s(-2,0)	r(1,0)-s(2,-1)
3+1x8+7		r(3,0)-s(1,-1)	s(-2,0)	r(3,1)-s(2,-1)	s(-2,0)	r(2,0)-s(2,-1)	s(-1,0)	r(2,1)-s(3,-1)
3+2x8+0	r(0,2)-s(-2,0)	s(2,-1)	s(-2,0)	r(3,0)-s(2,-1)	s(-1,0)	r(3,1)-s(3,-1)	s(-1,0)	r(2,0)-s(3,-1)
3+2x8+1	r(0,1)-s(-2,0)	s(2,-1)	r(0,2)-s(-1,0)	s(3,-1)	s(-1,0)	r(3,0)-s(3,-1)	s(0,0)	r(3,1)-s(4,-1)
•	•	•	r(0,1)-s(-1,0)	s(3,-1)	r(0,2)-s(0,0)	s(4,-1)	s(0,0)	r(3,0)-s(4,-1)
•	•	•	•	•	r(0,1)-s(0,0)	s(4,-1)	r(0,2)-s(1,0)	
3+3x8+0	s(2,0)	r(0,3)-s(-2,-1)	•	•	•	•	r(0,1)-s(1,0)	
3+3X8-1	s(2,0)	r(0,2)-s(-2,-1)	s(3,0)	r(0,3)-s(-1,-1)	•	•	•	•
•	•	•	s(3,0)	r(0,2)-s(-1,-1)	s(4,0)	r(0,3)-s(0,-1)	•	•
•	•	•	•	•	s(4,0)	r(0,2)-s(0,-1)		r(0,3)-s(1,-1)
3+3x8+6		r(3,3)-s(1,-1)		•		•		r(0,2)-s(1,-1)
3+3x8+7		r(3,2)-s(1,-1)	s(-2,2)	r(3,3)-s(2,-1)	s(-2,2)	•		•
3+4x8+0	r'(0,0)-s(-2,2)	s(2,1)	s(-2,2)	r(3,2)-s(2,-1)	s(-1,2)	r(3,3)-s(3,-1)		•
3+4X8-1	r(0,3)-s(-2,2)	s(2,1)	r'(0,0)-s(-1,2)	s(3,1)	s(-1,2)	r(3,2)-s(3,-1)	s(0,2)	r(3,3)-s(4,-1)
•	•		r(0,3)-s(-1,2)	s(3,1)	r'(0,0)-s(0,2)	s(4,1)	s(0,2)	r(3,2)-s(4,-1)
•	•		•		r(0,3)-s(0,2)	s(4,1)	r'(0,0)-s(1,2)	
3+4x8+6	r'(3,0)-s(1,2)		•		•		r(0,3)-s(1,2)	
3+4x8+7	r(2,3)-s(1,2)		r'(3,0)-s(2,2)		•		•	
3+5x8+0	s(2,2)	r'(0,1)-s(-2,3)	r(2,3)-s(2,2)		r'(3,0)-s(3,2)		•	
3+5x8+1	s(2,2)	r'(0,0)-s(-2,3)	s(3,2)	r'(0,1)-s(-1,3)	r(2,3)-s(3,2)		r'(3,0)-s(4,2)	
3+5x8+2	s(3,2)	r'(1,1)-s(-1,3)	s(3,2)	r'(0,0)-s(-1,3)	s(4,2)	r'(0,1)-s(0,3)	r(2,3)-s(4,2)	

그림 4. 선형 시스톨릭 어레이의 데이타 흐름도.

Fig. 4. Data flow of the Linear Systolic Array.

탐색영역의 짝수열 데이터 s(-2, -2)는 t=0와 t=1 두 클럭동안 path_0를 통하여 PE_3에 입력되고, 매 클럭마다에 한 프로세서씩 왼쪽으로 전달된다. s(4, -2)는 t=12, 13에서 path_0에 공급되고 두 클럭이 지난 후 t=16, 17에서는 탐색영역의 짝수열 데이터 s(-2, 0)가 path_0에 공급된다. 탐색영역의 홀수열 데이터 s(-2, -1)은 t=8과 t=9 두 클럭동안 path_1를 통하여 PE_3에 입력되고, 짝수열 데이터와 마찬가지로 매 클럭마다 한 프로세서씩 왼쪽으로 전달된다. s(4, -1)은 t=20, 21에서 path_1에 공급되고 두 클럭이 지난후 t=24, 25에서는 탐색영역의 홀수열 데이터 s(-2, 1)가 path_1에 공급된다. 기준블럭 데이터 r(0, 0)는 t=3에서 PE_0에 공급되고 매 클럭마다 한 프로세서씩 오른쪽으로 전달된다. r(1, 0)는 t=5에서, r(2, 0)는 t=7에서, r(3, 0)는 t=9에서 PE_0에 공급된다.

처음으로 기준블럭의 열이 공급되는 동안에는 하나의 기준블럭 열만이 PE_0에 공급되며, t=4, 6, 8, 10에서는 PE_0에 유효한 기준블럭 데이터가 공급되지 않는다. r(0, 1)은 t=11에 PE_0에 공급되고, t=12에서는 PE_0에 이미 공급되었던 r(0, 0)가 다시 입력된다. 이후 t=11~34까지는 현재 열의 데이터는 홀수 클럭에, 이전에 공급된 열의 데이터는 짝수 클럭에 교대로 입력된다.

T=33의 시점에서 기준블럭 R_1에대한 MAD(i-2, -2), $0 \le i < 4$의 연산을 위한 기준블럭 데이터의 공급은 완료되며, 각 프로세서는 연산이 완료된 MAD(i-2, -2), $0 \le i < 4$ 값을 순차적으로 왼쪽 프로세서에 전달한다. T=35이후의 홀수 클럭에서는 다음 기준블럭 R_2의 데이터 r(i, j)가 PE_0에 공급되며, 각 프로세서에서는 기준블럭 R_2에대한 MAD의 연산이 시작된다. T=36, 38....,42에서는 r(0, 3),...,r(2, 3)가 PE_0에 공급되어 기준블럭 R1에대한 MAD(i-2, -1), $0 \le i < 4$의 연산에 필요한 데이터의 공급이 완료되며, 각 프로세서는 연산이 완료된 MAD(i-2, -1), $0 \le i < 4$ 값을 순차적으로 왼쪽 프로세서에 전달한다. 이러한 과정은 한 슬라이스(slice)에 있는 모든 기준블럭 데이터와 탐색영역 데이터가 열 단위로 선형어레이의 모든 프로세서에 공급될 때까지 계속된다. 선형 어레이에 있는 4개의 프로세서는 각 기준블럭마다 2개씩 8개의 이동 벡터를 계산한다. 각 기준블럭의 MAD 값들에 대한 연산이 완료되는 시간은 아래와 같다.

T=3+3x8+6 부터 4 클럭동안
R_1:(-2, -2), (-1, -2), (0, -2), (1, -2).
T=3+4x8+7 부터 4 클럭동안
R_1:(-2, -1), (-1, -1), (0, -1), (1, -1).

T=3+7x8+6 부터 4 클럭동안
R_2:(-2, -2), (-1, -2), (0, -2), (1, -2).
T=3+8x8+7 부터 4 클럭동안
R_2:(-2, -1), (-1, -1), (0, -1), (1, -1).

.....

T=3+(4n-1)x8+6 부터 4 클럭동안
R_n:(-2, -2), (-1, -2), (0, -2), (1, -2).
T=3+(4n-1)x8+7 부터 4 클럭동안
R_n:(-2, -1), (-1, -1), (0, -1), (1, -1).

기준블럭의 크기가 NxN인 경우에 제안된 선형 어레이는 N 개의 프로세서를 가진다. 이 경우에도 각 프로세서는 모든 기준블럭에 대해 두 개의 이동 벡터에 대한 MAD 값들을 계산하므로 전체 선형 시스토릭 어레이는 2N개의 이동 벡터에 대한 MAD 값 의 연산을 수행한다. 따라서 크기가 N인 선형 시스토릭 어레이의 탐색범위는 2N 임을 알 수 있다.

본 장에서 제안한 선형 시스토릭 어레이에 공급되는 탐색영역과 기준블럭 데이터들은 두 클럭에 하나씩 입력된다. 따라서 매 클럭마다 새로운 데이터를 입력하는 기존의 방식들에 비해 요구하는 입력 대역폭이 낮다[9]~[13]. 또한 슬라이스에 있는 기준블럭 데이터와 탐색영역 데이터가 열단위로 한번씩만 선형 시스토릭 어레이에 입력되면 슬라이스에 있는 모든 기준블럭에 대해서 2N 개의 이동 벡터에 대한 MAD 연산이 수행되므로 이웃한 기준블럭 연산시 반복해서 탐색영역 데이터를 새로이 입력하는 방식들에 비해 데이터의 활용도(utilization)가 높으며 데이터의 공급과 제어가 쉽다[9][11]. 처음에 탐색영역 데이터가 선형 어레이를 채우는데 필요한 시스템 초기화 과정을 제외하면 각 프로세서가 100% 연산 효율을 갖는다.

Ⅲ. N x 2p의 탐색 범위를 갖는 이동 예측기

본 장에서는 Ⅱ 장에서 제안된 선형 시스토릭 어레

이를 이용하여 기준블럭의 크기가 NxN이고 탐색범위의 크기가 Nx2p인 이동 예측기를 구성하는 방법을 설명한다. 그림 5에 탐색범위가 Nx2p인 이동 예측기가 나타나있다. 이이동 예측기는 선형 시스토릭 어레이를 p개 사용하며, 각 선형 어레이는 N개의 프로세서를 가지고 있다. P 개의 선형 시스토릭 어레이는 수직 방향으로 나열되어 있다. 위에서 i 번째의 선형 시스토릭 어레이를 PIPE i로 나타내기로 한다. PIPE i에 공급되는 기준블럭 데이터는 PIPE 0에 공급되는 기준블록 데이터가 2Nxi 개의 지연 레지스터를 통하여 전달된다. 각 지연 레지스터는 프로세서의 반의 속도로 동작하며, 따라서 2Nxi 개의 지연 레지스터는 4Nxi 클럭만큼 기준블럭 데이터를 지연시키는 역할을 한다. 기준블럭 데이터는 좌 상에서와 마찬가지로 현재의 열과 2N 클럭만큼 지연된 이전의 열이 멀티플렉서를 통해서 교대로 각 파이프에 공급된다. 탐색영역 데이터는 path 0와 path 1을 통하여 모든 선형 시스토릭 어레이에 공통으로 공급된다. 이후 본 논문에서는 Nx2p 탐색영역에 있는 이동 벡터들을 u(0, 0) ,...,u(N-1, 2p-1)로 나타내고, 각 이동벡터의 MAD 값을 MAD(0, 0),...,MAD(N-1, 2p-1)로 나타내기로 한다. 제안한 이동 예측기의 PIPE i에서는 슬라이스안에 있는 모든 기준블럭에 대하여 MAD(i, 0), MAD(i, 1), $0 \le i < N$를 계산한다.

그림 5의 이동 예측기에 기준블럭과 탐색영역 데이터를 공급하는 방법은 앞 장에서 한개의 선형 시스토릭 어레이에 데이터를 공급하는 방법과 같다. 탐색영역 데이터는 연속된 2 클럭마다 모든 파이프의 PE_{N-1} 프로세서에 공급된다. 공급된 탐색영역 데이터는 각 파이프의 path 0 또는 path 1을 통하여 매 클럭마다 왼쪽의 프로세서로 전달된다. 탐색영역의 짝수열 데이터는 path 0을 통하여 공급되며, 탐색영역의 홀수열 데이터는 path 1을 통하여 공급된다. 기준블럭 데이터는 탐색영역 데이터가 공급된지 N 클럭만에 PIPE 0의 PE_0에 공급된다. 즉, 슬라이스의 첫 기준블럭 R_1의 좌측상단 화소 데이터 r(0, 0)는 $t_0 = (N-1) + 0x2N$에 PIPE 0의 PE_0에 공급되고, 매 클럭마다 한 프로세서씩 오른쪽으로 전달된다. R_1의 r(1, 0)는 $t = t_0 + 2$에서, r(2, 0)는 $t = t_0 + 4$에서,..., r(N-1, 0)는 $t = t_0 + (2N-2)$에서 PIPE 0의 PE_0에 공급된다. 처음으로 이동 예측기에 기준블럭의 열 r(i, 0), $0 \le i < N$가 공급되는 동안에는 $t = t_0 + 1$, $t = t_0 + 3$,..., $t = t_0 + (2N-1)$에서는 PIPE 0에 유효한 기준블럭 데이터가 공급되지 않는다. r(0, 1)은 $t_1 = (N-1) + 1x2N$에서 PIPE_0의 PE_0에 공급되고 $t = t_1 + 1$에서는 이미 공급되었던 r(0, 0)가 PIPE 0에 다시 공급된다. 이후 현재 열의 데이터는 홀수 클럭에, 이전에 공급된 열의 데이터는 짝수 클럭에 교대로 공급된다.

$t_{u(0,\ 0)} = (N-1) + (N-1)x2N + (2N-2)$의 시점에서 r(N-1, N-1)이 공급되어 PIPE 0의 PE_0에서 처음 기준블럭 R_1의 MAD(0, 0)의 계산이 완료된다. 이후 N 클럭동안 R_1의 이동벡터들의 MAD(i, 0), $0 \le i < N$가 PIPE 0의 각 프로세서에서 순차적으로 계산 완료된다. 그리고, R_1의 r(N-1, N-1)이 공급된후 홀수 클럭에는 두번째 기준블럭 R_2의 좌측상단 화소 데이터 r(0, 0)가 공급되어 R_2의 이동베터들의 MAD 값을 위한 연산이 계속해서 이루어진다. $tu_{u(0,\ 1)} = (N-1) + Nx2N + (2N-1)$의 시점에서는 이미 공급된 r(N-1, N-1)가 PIPE 0의 PE_0에 공급되어 R_1의 MAD(0, 1) 계산이 완료되며, 이후 N 클럭동안 R_1의 이동벡터들의 MAD (i, 1), $0 \le i < N$가 계산 완료된다. 이러한 과정은 슬라이스안의 모든 데이터가 PIPE 0에 공급될동안 계속되어 PIPE 0에서는 슬라이스안의 모든 기준블럭에 대한 MAD(i, 0), MAD(i, 1), $0 \le i < N$를 계산한다.

인접한 두 파이프에 공급되는 기준블럭 데이터 상호사이에는 4N 클럭의 지연 버퍼가 있으므로, PIPE i에 기준블럭의 j 번째 열이 공급될 때 PIPE (i + 1)에는 기준블럭의 (j-2) 번째 열이 공급된다. 따라서 PIPE i의 PEk에서 MAD(k, 2i)와 MAD(k, 2i-1)을 계산할 때 PIPE (i + 1)의 PE_k에서는 MAD(k, 2i + 2)와 MAD(k, 2i + 1)을 계산하게 된다.

한 슬라이스안에 있는 각 기준블럭의 이동벡터 u (0, 0)에 대한 MAD(0, 0)가 계산 완료되는 시점은 식 (2)와 같다.

$$
\begin{aligned}
&R_1 : t_{u(0,\ 0)} = (N-1) + (N-1)x2N + (2N-2),\\
&R_2 : t_{u(0,\ 0)} = R_1 : t_{u(0,\ 0)} + \{(N-1)x2N + (2N-2)\},\\
&\qquad = (N-1) + (2N^2-2)x2, \qquad (2)\\
&\cdots\cdots\\
&R_i : t_{u(0,\ 0)} = (N-1) + (2N^2-2)\ x\ i.
\end{aligned}
$$

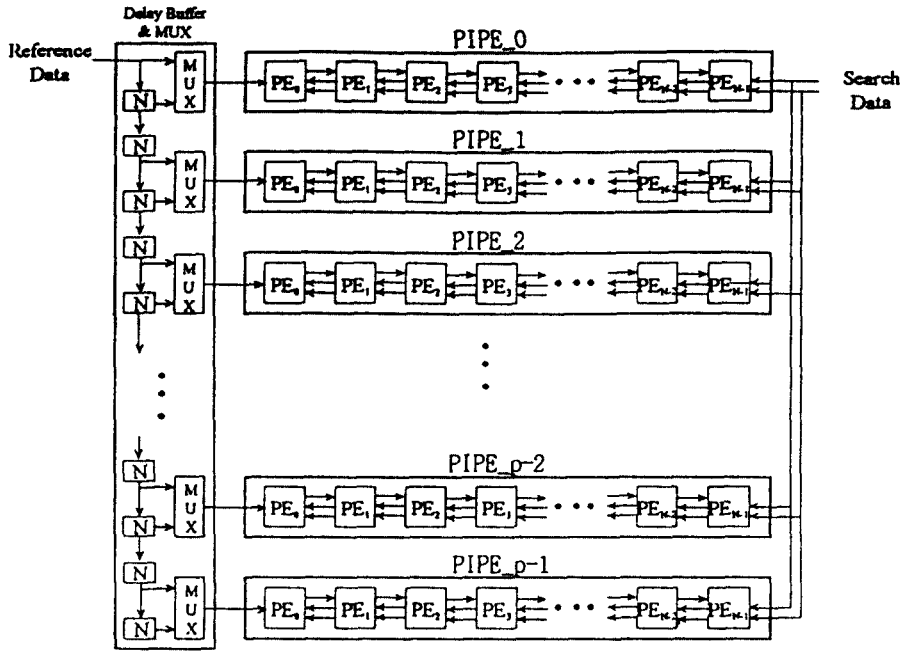


그림 5. N x 2p의 탐색범위를 갖는 이동 예측기
Fig. 5. Motion Estimator with N x2p Search range.

R1의 r(0, 0)값이 PIPE_p-1의 PE_{N-1}에 장입되는 시점은 (N-1) + {(p-1)x4N + (N-1)}이 된다. 한 슬라이스 안에 있는 각 기준블럭의 마지막 이동벡터 u(N-1, 2p-1)에 대한 MAD(N-1, 2p-1)이 계산 완료되는 시점을 식으로 나타내면 식 (3)과 같다.

$$R_1 : tu_{u(N-1,\,2p-1)} = (N-1) + \{(p-1)x4N + (N-1)\} + \{(N-1)x2N + (2N-2)\} + (N+1),$$
$$R_2 : tu_{u(N-1,\,2p-1)} = (N-1) + \{(p-1)x4N + (N-1)\} + \{(N-1)x2N + (2N-2)\}x2 + (N+1),$$
$$= (N-1) + (4pN-3N-1) + (2N^2-2)x2 + (N+1), \quad (3)$$

.....

$$R_i : tu_{u(N-1,\,2p-1)} = (N-1) + (4pN-3N-1) + (2N^2-2)xi + (N+1),$$
$$= 4pN + (2N^2-2)xi - N - 1.$$

한 프레임의 크기가 $FvxF_h$이고, 기준블럭의 크기가 NxN이며, 한 기준블럭이 Nx2p개의 이동벡터를 가질경우에 대해서, 그림 5의 이동 예측기로 한 기준블럭의 최종 이동벡터를 계산하는데 소요되는 클럭 사이클 수와 전체 프레임을 처리하는데 소요되는 클럭 사이클 수는 식 (3)으로부터 구할 수 있으며, 각각 식 (4)와 식 (5)와 같다.

$$\#\ \text{of Clock Cycle/block} = [4pN + (2N^2-2)xF_v/N - N - 1]xN/F_v. \quad (4)$$
$$\#\ \text{of Clock Cycle/Frame} = [4pN + (2N^2-2)xF_v/N - N - 1]xF_h/N. \quad (5)$$

Ⅳ. kNx2p의 탐색범위를 갖는 이동 예측기

본 장에서는 Ⅲ장에서 설계한 탐색범위 Nx2p인 이동 예측기를 이용하여 탐색범위가 kNx2p인 이동 예측기를 설계하는 방법에 대해 설명한다. 설명의 편의를 위하여 k = 2인 경우를 예로 들기로 한다.

그림 6에 Nx2p의 이동 예측기를 이용하여 2Nx2p의 탐색범위를 갖는 이동 예측기가 나타나 있다. 2Nx2p 이동 예측기에는 (0~N-1)x2p 탐색범위를위한 Nx2p 이동 예측기(ME_0)와 (N~2N-1)x2p 탐색범위를 위한 Nx2p 이동 예측기(ME_1)가 있다. 또한, 탐색영역

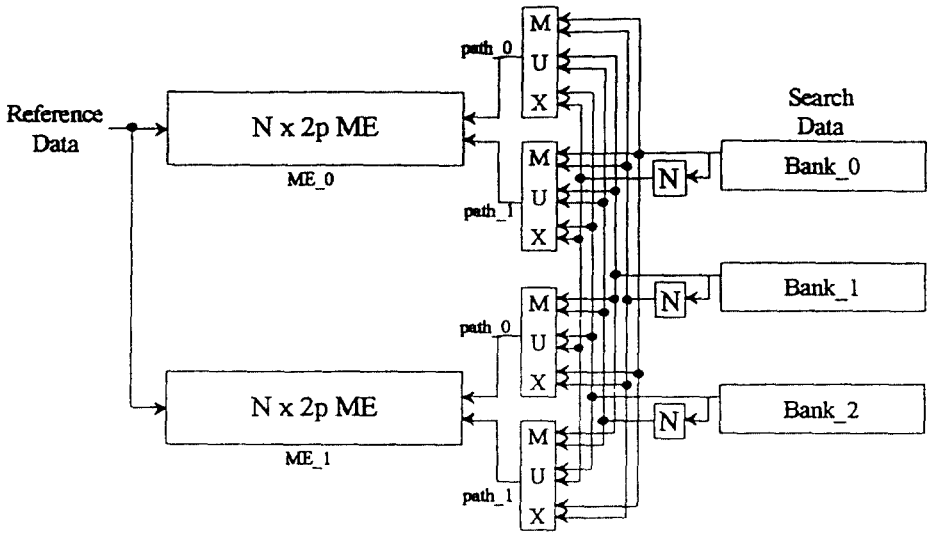


그림 6. 2 x 2p 이동 예측기 구성도.
Fig. 6. Block Diagram of 2N x 2p Motion Estimator

데이터를 Nx2p 이동 예측기의 path 0와 path 1에 공급하기위한 지연버퍼와 멀티플렉서, 그리고 탐색영역 데이터를 저장하기위한 세개의 메모리 뱅크가 있다. 그림 6의 이동 예측기에 탐색영역 데이터를 공급하는 규칙은 아래와 같다.

1. 탐색영역 데이터는 N개의 행을 하나의 그룹 단위로 나누며, 나누어진 세개의 그룹(그룹 0, 그룹 1, 그룹 2)은 세개의 메모리 뱅크에 저장된다.
2. 이동 예측기 ME_0에는 그룹 0와 그룹 1 탐색영역 데이터가 입력되게 하고, 이동 예측기 ME_1에는 그룹 1과 그룹 2 탐색영역 데이터가 입력되도록 지연버퍼 또는 멀티플렉서가 동작한다.
3. 각 이동 예측기의 path 0와 path 1에 인가된 멀티플렉서는 각 뱅크의 홀수번째 열 데이터가 path 0을 통하여 공급되게하고, 각 뱅크의 짝수번째 열 데이터는 path 1을 통하여 이동 예측기에 공급되도록 동작한다.
4. 각 뱅크의 탐색영역 데이터는 매 2 클럭에 한 데이터씩 열순으로 공급된다.

위의 규칙 1에서 탐색영역을 그룹 단위로 나눔으로써, 그룹 1의 탐색영역 데이터를 ME_0와 ME_1의 인산에 중복하여 사용하는것이 가능하여 한 그룹만큼의 탐색영역 데이터를 새로이 입력하지 않아 데이터의 활용도가 높아진다. 또한, 다음 슬라이스의 기준블럭에대한 탐색영역 데이터의 입력시 그룹 1과 그룹 2의 탐색영역 데이터는 각각 다음 슬라이스의 기준블럭에 대한 그룹 0와 그룹 1 탐색영역 데이터가 되며, 다음 슬라이스의 기준블럭에 대한 그룹 2 탐색영역 데이터만 입력하여 시스템의 입력 데이터율이 낮게된다.

이후 그림 6 시스템의 동작 설명은 그룹 0, 그룹 1, 그룹 2 탐색영역 데이터가 각각 bank 0, bank 1, bank 2에 있을경우에 대한 것이다. t=0에 bank 0, bank 1의 첫번째 열 데이터가 매 2 클럭에 한 데이터씩 멀티플렉서를 통해 각각 ME_0와 ME_1의 path 0로 공급된다. 이때 path 1을 통해서는 어떤값도 공급되지 않는다. t=2N-1에 각 뱅크의 첫번째 열이 공급이 완료되며, 두번째 열이 공급된다. 이 시점에서 ME_0의 path 0에는 bank 1의 첫번째 열 데이터가 지연버퍼를 통하여 공급되며, path 1을 통해서는 bank 0의 두번째 열 데이터가 공급된다. ME_1에서도 역시 path 0에는 bank 2의 첫번째 열 데이터가 지연 버퍼를 통하여 공급되며, path 1을 통해서는 bank 1의 두번째 열 데이터가 공급된다. 각 이동 예측기의 path 0와 path 1에 인가된 멀티플렉서는 각 뱅크의 홀수번

째 열 데이터가 path_0를 통하여 공급되게하고, 각 뱅크의 짝수번째 열 데이터는 path_1을 통하여 이동 예측기에 공급되도록 동작한다. 이같은 탐색영역 데이터 공급은 Ⅱ장에서 설명한 선형 어레이에 탐색영역 데이터 공급을위한 두번째 규칙을 만족한다.

다음 슬라이스의 기준블럭을 위한 그룹_0, 그룹_1, 그룹_2 탐색영역 데이터는 각각 bank_1, bank_2, bank_0에 저장된다.

기준블럭 데이터는 Ⅱ장에서 설명한 기준블럭 데이터의 공급 규칙에 준하여 ME_0와 ME_1에 공급된다. ME_0와 ME_1에서의 동작은 공급되는 기준블럭 데이터와 탐색영역 데이터에 대해서 서로 독립적으로 동작한다.

ME_0의 PIPE_0 PE_0에서는 슬라이스내에 있는 모든 기준블럭의 MAD(0, 0)와 MAD(0, 1)이 계산되며, ME_1의 PIPE_0 PE_0에서는 ME_0에 공급되는 같은 기준블럭의 MAD(N, 0)와 MAD(N, 1)이 계산된다. 즉, 2Nx2p 이동 예측기 전체적으로 ME_0에서는 슬라이스내에 있는 모든 기준블럭의 MAD(i, j), $0 \le i < N-1$, $0 \le j < 2p-1$이 계산되며, ME_1에서는 MAD(i + N, j), $0 \le i < N-1$, $0 \le j < 2p-1$이 같은 시점에 계산된다.

2Nx2p 이동 예측기가 한 기준블럭의 이동벡터를 구하는데 소요되는 클럭 사이클 수와, 한 프레임을 처리하는데 소요되는 클럭 사이클 수는 탐색범위가 Nx2p일때와 같다.

Ⅴ. 성능 분석

본 장에서는 본 논문에서 제안하는 선형 시스토릭 어레이를 사용한 이동 예측기의 성능을 분석하였다.

표 1에는 기준블럭의 변화와 탐색 범위의 변화에 따른 전체 프로세서 갯수, 입력 핀의 갯수, 한 기준블럭에대한 이동벡터를 계산하는데 소요되는 연산 사이클 수와 초당 처리가능한 프레임 수를 나타내었다.

표 1에서 블럭당 소요 클럭수와 초당 처리 가능한 프레임 수는 입력 비디오 신호가 720x480이며, 시스템 동작 클럭이 40 Mhz(25 ns)일경우에 대한 값이다. 표 1에서 보인바와 같이 제안한 선형 시스토릭 어레이 이동 예측기는 작은 입력 핀수를 가지며, 기준블럭의 크기변화 또는 탐색범위의 변화에대해서 입력 핀 수의 변화가 작다. 한 기준블럭의 이동벡터를 구하는데 소요되는 클럭 사이클 수는 식(4)에의해 구해지며, 매 초당 처리 가능한 화면 수는 식(5)에의해서 구해진다.

표 1에서 3번째와 4번째 예는 같은 기준블럭 크기와 같은 탐색범위를 가지며, 3번째 예는 2Nx2p 이동 예측기 구조를 사용하였고, 4번째 예는 Nx2p 이동 예측기를 사용하여 구현한 것이다. 그 결과 사용 프로세서의 수는 반으로 줄고, 수행 속도는 2배로 증가한다.

표 2에서는 여러 크기의 프레임을 처리하기위한 클럭 사이클 수, 그리고 각각에 대한 실시간 처리에 요구되는 동작 속도가 나타나 있다. 실시간 처리 계산은 1초당 30 프레임의 처리에 대한것이다. 최대 이동벡터값이 -48~47인 예는 이웃한 프레임 사이의 이동범위가 -16~15이며, 압축방법이 I-B-B-P 형태의 화면 구성을 가지는 MPEG-I과 MPEG_II의경우 입력순으로 I 프레임 이후 P 프레임이 입력되므로 세 프레임 간격이 생긴 경우를 처리한 것이다. 본 논문에서 제안한 선형 시스토릭 어레이 이동 예측기는 동작속도가 23.5Mhz일경우 MPEG-II의 실시간 처리를 위한 30 Frame/sec를 만족한다.

표 3에서는 다른 알고리즘과의 성능 비교가 나타나 있다[14].

표 3에서 클럭 수는 기준 블럭이 16x16이고, 탐색범위가 -16~15 일경우이며, Frame에 관한 데이터는

표 1. 선형 시스토릭 어레이 이동 예측기 성능분석(I)

Ref. Block Size	Search Range	# of Input	# of Input data port	# of Cycle/block	Frame/Sec.
8x8	−8~7	128	4	129	57
16x16	−8~7	128	3	521	56
16x16	−16~15	512	4	533	55
16x16	−16~15	256	3	1,065	27

표 2. 선형 시스토릭 어레이 이동 예측기 성능분석(II)

TYPE	Frame size	Ref. block size	Search Range	# of Clock cycle/frame	Req. Speed for 30 f/s
QCIF	176 x 144	16 x 16	-16 ~ 15	59,598	1.8 MHz
CIF	352 x 288	16 x 16	-16 ~ 15	220,086	6.7 MHz
SIF	320 x 240	16 x 16	-48 ~ 47	198,825	6 MHz
ITU-R 601	720 x 480	16 x 16	-48 ~ 47	780,150	23.5 MHz

표 3. 선형 시스토릭 어레이 이동예측기 성능분석(III)

TYPE	INPUT Pin Count	# of Clock Cycles /Block	# of Clock Cycles /Frame	# of PE
Komarek	16	1,584	2,566,080	528
Heieh & Lin	2	2,310	3,742,200	256
Yang	6	4,096	6,635,520	64
Ours	4	522	918,759	128

입력 비디오 신호가 CCIR625 x 720에 대한것이다. 표 3에서 본 논문에서 제안한 선형 시스토릭 어레이 이동 예측기는 Komarek, Yang의 알고리즘 보다는 적은 입력 핀의 수를 가지며, Komarek, Heieh & Lin의 알고리즘 보다는 적은 PE의 갯수를 가지며 빠른 처리가 가능함을 볼 수 있다.

VI. 결 론

본 논문에서는 선형 시스토릭 어레이에 기초한 완전탐색 블럭정합 이동 예측기의 새로운 구조를 제안하였다. 제안된 이동 예측기에 적용된 선형 시스토릭 어레이 구조는 입력 데이터뿐만아니라 제어신호까지 레지스터를 통한 파이프라인 형태로 내부 프로세서에 공급함으로써 버스의 사용을 제한하여 고속의 동작이 가능하다. 시스템 구조의 기본이되는 연산기는 매 클럭 새로운 입력 데이터에대한 연산이 이루어져 100%의 효율을 가진다. 또한 외부 입력 데이터가 프로세서 동작속도의 절반의 속도로 입력되게 함으로써 입력 데이터 대역폭이 매우낮으며, 블럭크기의 변화와 탐색범위의 증가에 대해 기존의 이동 예측기에 비해 요구되는 프로세서의 수와 입력 데이터 대역폭 면 그리고 입력 핀 수에서 매우 좋은 확장성을 가지고 있다. 마지막으로 제안된 구조는 선형 시스토릭 어레이의 상점인 규칙성이 있어 그 구조가 간단하며, 모듈 설계가 쉽고, 제어가 간단하여 VLSI로 구현하기에 적당하다.

참 고 문 헌

1. "Video Codec for Audiovisual Services at p x 64 kbit/s", ITU CCITT Recommendation H. 261, 1990.
2. "Coding of Moving Pictures and Associated Audio", ISO/IEC JTC1/SC29/WG11 MPEG 92, March, 1992.
3. "Coding of Moving Pictures and Associated Audio", ISO/IEC JTC1/SC29/WG11 N0702rev, Recomm. H. 262, March 1994.
4. F. Thomson Leighton, "Introduction to Parallel Algorithms and Architectures", Morgan Kaufmann Publishers, Inc., 1992
5. M. N. Pettigrew, Vijay K. Madisetti, "Progress in Algorithms and VLSI Architectures for Motion Estimation and Compensation in Video Compression", National Center of Excellence in DSP School of Electrical Engineering, July 29, 1993.
6. E. D. Frimout, J. N. Driessen, Ed F. Deprettere, "Parallel Architecture for a Pel-Recursive Motion Estimation Algorithm", IEEE Trans. on circuits and systems for video tech., Vol. 2, No. 2, pp. 159~168, 1992.
7. L. G. Chen, W. T. Chen, Y. S. Jehng, T. D. Chiueh, "An Efficient Parallel Motion Estimation Algorithm for Digital Image Processing", IEEE

Trans. on circuits and systems for Video Tech., Vol. 1, No. 4, pp. 378~385, 1991.

8. L. W. Lee, J. F. Wang, J. Y. Lee, J. D. Shie, "Dynamic Search-Window Adjustment and Interlaced Search for Block-Matching Algorithm", IEEE Trans. on circuits and systems for Video Tech., Vol. 3, No. 1, pp. 85~87, 1993.
9. K. M. Yang, M. T. Sun, L. Wu, "A Family of VLSI Designs for the Motion Compensation Block-Matching Algorithm", IEEE Trans. on circuits ans systems, Vol. 36, No. 10, pp. 1317~1325, Oct. 1989.
10. T. Komarek, P. Pirsch, "Array Architectures for Block Matching Algorithms", IEEE Trans. on circuits and system, Vol. 36, No. 10, pp. 1301~1308, Oct. 1989.
11. Hsieh, T. P. Lin, "VLSI Architecture for Block-Matching Motion Estimation Algorithm", IEEE Trans. on circuits and systems for Video Tech., Vol. 2, No. 2, pp. 169~175, June 1992.
12. Ruetz, P. Tong, D. Bailey, D. A. Luthi, P. H. Ang, "A High-Performance Full-Motion Video Compression Chip Set", IEEE Trans. on circuits and system for video tech., Vol. 2, No. 2, pp. 111~121, June 1992.
13. Wu, D. K Yeh, "A VLSI Motion Estimator for Video Image Compression", IEEE Trans. on Consummer electromics, Vol. 39, No. 4, pp. 837~846, Aug. 1993.
14. Hangu Yeo, Yu Hen Hu, "A Novel Moduar Systolic Array Architecture for Full-search Block Matching Motion Estimation", IEEE ICASSP-95, pp. 3303~3306, 1995.

金 起 賢(Ki Hyun Kim) 정회원

1989년 2월 : 경북대학교 전자공학과 졸업(공학사)

1991년 2월 : 경북대학교 대학원 컴퓨터공학과 졸업(공학석사)

1991년 2월~현재 : 한국전자통신연구소 컴퓨터연구단 미디어연구실, 선임연구원

※주관심 분야 : ASIC 설계, VLSI 구조, 영상처리, 병렬처리

金 起 徹(Kichul Kim) 정회원

1982년 2월 : 서울대학교 전기공학과 졸업(공학사)

1984년 2월 : 서울대학교 대학원 전기공학과 졸업(공학석사)

1991년 8월 : University of Southern California 전기공학과 졸업(Ph. D)

1984년 3월~1994년 2월 : 한국전자통신연구소 선임연구원

1994년 3월~현재 : 서울시립대학교 반도체공학과 전임강사

※주관심 분야 : ASIC 설계, VLSI 구조, 병렬처리