논문 2009-46SD-6-2

H.264 움직임추정에서 고속 2D PE 아키텍처의 메모리대역폭 개선을 위한 4-방향 검색윈도우

(4-way Search Window for Improving The Memory Bandwidth of High-performance 2D PE Architecture in H.264 Motion Estimation)

고 병 수*, 공 진 흥**

(Byung Soo Ko and Jin-Hyeung Kong)

요 약

본 논문에서는 H.264 움직임추정의 고속처리를 위하여 2D PE 아키텍처의 메모리 대역폭을 개선할 수 있는 새로운 4-방향 검색윈도우를 설계 및 구현하였다. 기존의 2D PE 아키텍처는 메모리 대역폭을 줄이기 위하여 스캔경로 내에서 인접한 검색윈도우간 중복되는 데이터를 재사용하였으나, 본 연구에서는 재사용을 증대시키기 위하여 인접한 스캔경로 간의 검색윈도우에 대해서도 재사용할 수 있는 방법을 제안한다. 이를 위해서 검색윈도우를 하나의 스캔경로 내에서 래스터 및 사행 스캐닝을 수행하는 기존 방식을 개선하여, 인접한 복수 스캔경로를 4방향(상, 하, 좌, 우)으로 스캐닝하면서 이동할 수 있는 검색윈도우를 설계하였다. 기존 검색윈도우가 제한적인 데이터 재사용으로 7.7~11회 정도의 중복적인 검색(redundancy access factor)을 요구하는데 비하여, 제안된 4-방향 검색윈도우는 3.1/1.4회 정도로 중복검색을 감소시킨 성능을 보인다. 이에 따라서 4-방향 검색윈도우는 기존의 1-방향 검색윈도우에 비하여 70%, 3-방향 검색윈도우에 비하여 60%/81%의 메모리 대역폭 개선 효과를 가져올 수 있게 된다. 제안된 4-방향 검색윈도우의 H.264 정수화소 움직임추정 아키텍처는 절대차분 연산을 위한 16×16의 2D PE어레이와 인접 스캔경로 간 검색윈도우 데이터를 재사용하기 위한 5×16의 RE어레이로 구성되어 있다. 2D PE어레이는 스캔방향에 따라 상/하 양방향으로 참조데이터를 입력받을 수 있으며, 인접한 복수 스캔경로들의 데이터 재사용을 위한 RE 어레이가 2D PE어레이와 함께 좌/우 양방향으로 로테이트가 가능하도록 구성되어 있다. 4방향 검색윈도우는 Magnachip 0.18um공정으로 구현되어, H.264 움직임추정 메모리대역폭을 개선하여 2D PE 아키텍처 사양 참조 프레임 1장, 검색영역 48×48, 매크로블록 16×16의 HD영상(1280×720)을 149.25MHz에서 실시간처리하는 성능을 보였다.

Abstract

In this paper, a new 4-way search window is designed for the high-performance 2D PE architecture in H.264 Motion Estimation(ME) to improve the memory bandwidth. While existing 2D PE architectures reuse the overlapped data of adjacent search windows scanned in 1 or 3-way, the new window utilizes the overlapped data of adjacent search windows as well as adjacent multiple scanning (window) paths to enhance the reusage of retrieved search window data. In order to scan adjacent windows and multiple paths instead of single raster and zigzag scanning of adjacent windows, bidirectional row and column window scanning results in the 4-way(up, down, left, right) search window. The proposed 4-way search window could improve the reuse of overlapped window data to reduce the redundancy access factor by 3.1, though the 1/3-way search window redundantly requires 7.7~11 times of data retrieval. Thus, the new 4-way search window scheme enhances the memory bandwidth by 70~58% compared with 1/3-way search window. The 2D PE architecture in H.264 ME for 4-way search window consists of 16×16 pe array, computing the absolute difference between current and reference frames, and 5×16 reusage array, storing the overlapped data of adjacent search windows and multiple scanning paths. The reference data could be loaded upward and downward into the new 2D PE depending on scanning direction, and the reusage array is combined with the pe array rotating left as well as right to utilize the overlapped data of adjacent multiple scan paths. In experiments, the new implementation of 4-way search window on Magnachip 0.18um could deal with the HD(1280×720) video of 1 reference frame, 48×48 search area and 16×16 macroblock by 30fps at 149.25MHz.

Keywords : H.264, motion estimation, 2D PE, 4-way, scanning, memory bandwidth, search window

* 정회원, ** 평생회원, 광운대학교 컴퓨터공학과
(Department of Computer Engineering, Kwangwoon University)

※ "본 논문은 2008년도 광운대학교 교내학술연구비 지원에 의해 연구되었으며, 2009년도 「서울시 산학연 협력사업」과ETRI 「IT SoC 핵심설계인력양성사업」의 결과이며, 반도체 설계교육센터(IDEC)의 지원으로 이루어졌습니다."

접수일자: 2009년1월7일, 수정완료일: 2009년6월1일

I. 서 론

H.264 움직임추정은 부호화기 전체에서 약 95% 이상의 연산 및 메모리 접근(access)을 차지하고 있다[1]. 따라서 H.264 움직임추정을 실시간으로 처리하기 위하여 고속의 병렬 연산 구조에 관한 연구[2~3]들과 병행하여, 병렬 연산 구조의 메모리 대역폭을 줄이기 위한 중간버퍼관리[4~5]와 검색윈도우관리[6~10]등이 제안되고 있다. 기존 중간버퍼관리에서는 검색영역(M×M)에 대한 데이터 재사용을 높이는 메모리관리기법으로 중간버퍼에 대한 외부메모리의 대역폭 문제를 해결하고자 한다. 검색윈도우관리(N×N)에서는 검색윈도우에 대한 데이터 재사용을 높이는 부가레지스터 관리기법으로써 2D PE에 대한 대역폭 문제를 해결하고자 하였다. 실제로 움직임추정을 위한 현재값 및 참조값의 2차원적 차분값을 2D PE 구조가 고속 연산하기 위해서 각각의 pe셀이 요구하는 참조데이터를 공급받아야 하는데 이는 매우 큰 메모리 밴드폭을 필요로 하게 된다. 이같은 2D PE 구조가 요구하는 밴드폭을 줄이기 위해서 외부 메모리와 2D PE 사이에 재사용중간버퍼인 SAM (Search Area Memory)을 두어 2D PE의 각 pe셀 내에 참조데이터를 공급하거나[4~5], 2D PE 구조 내 pe셀과 쉬프트레지스터를 결합시켜 데이터 이동 및 공유가 가능 하도록 설계하여 참조데이터를 재사용[6~10]하고 있다. 재사용중간버퍼 SAM 구조는 2D PE에서 각각의 pe셀마다 움직임벡터를 찾는 검색범위 만큼의 참조데이터를 저장하기 때문에, 매우 큰 메모리 용량을 포함시켜야 하고 따라서 저장 및 검색 등 메모리 관리 오버헤드가 커서 동작속도를 높일 수 없는 구조적 한계를 갖는다. 반면에 쉬프트레지스터 구조는 인접 pe셀과 참조데이터를 공유해서 메모리 용량을 각각의 pe셀에서 필요한 하나의 참조데이터크기로 제한시킬 수 있다. 또한 인접 pe셀 간의 연결네트워크에 따라서 공유 및 재사용 정도가 다르게 되고 메모리 밴드폭도 결정된다. 실제로 2D PE 내부에서 참조데이터를 재사용하기 위한 쉬프트레지스터 기반 인접 pe셀간의 네트워크에 대한 연구는 참조데이터 재사용을 위한 1-방향/3-방향 검색윈도우[6~10]구조를 제안하고 있다.

1-방향 검색윈도우에 관한 연구는 참조프레임의 검색영역 내에서 검색윈도우를 래스터(raster) 방식으로 스캐닝하며, 스캔 진행방향의 인접한 후보 블록 간 참조데이터를 재사용한다[6~8]. 이러한 1-방향 검색윈도우는 재사용을 적용하지 않았을 경우와 비교해서 91% 정도의 메모리대역폭을 감소시켰다. 1-방향 검색윈도우에서 스캔경로가 바뀔 때 마다 발생하는 초기 지연시간 및 데이터 재사용 문제를 개선하기 위하여, 스캔경로가 바뀌는 부분에서 다음 스캔경로가 연속적으로 재사용될 수 있도록 사행(snake) 형태의 3-방향 스캔이 제안되었다[9~10]. 이같은 3-방향 검색윈도우는 스캔경로의 시작과 끝부분에서 재사용을 통하여 1-방향 검색윈도우에 비하여 메모리 대역폭을 30% 정도 개선하였다. 본 연구에서는 스캔경로의 시작과 끝지점에서의 참조데이터 재사용뿐만 아니라, 스캔경로 중간에서 인접한 스캔경로와의 연속적인 데이터 재사용을 가능하게 하고자 한다. 이를 위하여 스캔경로 중간에서 검색윈도우의 스캐닝(위쪽, 아래쪽)을 스캔진행방향의 직선으로 진행하면서, 인접한 스캔경로를 포함하여 (왼쪽, 오른쪽)으로 이동할 수 있도록 하였다. 따라서 인접한 스캔경로의 검색윈도우 데이터를 재사용하기 위해서 검색윈도우가 스캔 진행방향(위쪽, 아래쪽) 뿐만 아니라 인접스캔경로(왼쪽, 오른쪽)를 포함한 4-방향으로 이동하는 방식으로 동작을 설계하였다. 실제로 4-방향 검색윈도우는 스캔경로의 데이터에 대한 재사용뿐만 아니라, 인접한 다수 스캔경로에 대한 재사용을 가능하게 해서 참조데이터의 재사용을 2차원적으로 확대시키게 된다. 제안된 4-방향 검색윈도우의 메모리 밴드폭과 데이터 재사용율, 쉬프트레지스터 그리고 검색의 효율성 등을 기존 1-방향/3-방향 검색윈도우와 비교하여 정량적 성능 개선효과를 증명하고자 한다.

본 논문에서 제안하는 4-방향 검색윈도우를 고속 H.264 움직임추정 2D PE에 적용시켜 최소의 메모리대역폭을 설계 및 구현하였다. 4-방향 검색을 하는 움직임추정기는 2D PE어레이, RE(Reuse Element)어레이, SAD(Sum of Absolute Difference)연산기로 구성되어 있다. 2D PE어레이는 검색윈도우 단위로 현재/참조 데이터를 입력받아 절대차분 연산을 수행하며, RE어레이와 함께 4-방향 쉬프팅을 지원하는 연결네트워크를 구성한다. RE어레이는 인접 스캔경로의 검색윈도우 데이터 재사용을 위한 쉬프트 레지스터 배열로서 인접경로 및 방향전환 RE어레이로 나누어진다. 스캔경로가 아래쪽(위쪽)으로 향할 때는 2D PE어레이 및 RE어레이의 데이터가 함께 한행씩 위쪽(아래쪽)으로 쉬프트 되면서 외부메모리로 부터 한행의 새로운 데이터를 아래쪽(위쪽)으로 부터 입력받는다. 또한 스캔경로가 왼쪽

(오른쪽)으로 향할 때는 2D PE어레이 및 RE어레이의 데이터가 한 열씩 오른쪽(왼쪽)으로 회전(rotate) 되도록 pe셀간의 연결 네트워크가 구성되어 있다. 이때 인접스캔경로를 포함하는 2차원 검색과정을 위해서 인접경로 RE어레이와 2D PE어레이가 연결되고, 방향전환 과정에서는 방향전환 RE어레이까지 재사용 연결네트워크를 구성하게 된다.

본 논문의 구성은 Ⅱ장에서 메모리 대역폭 개선을 위한 연구를 분석하고 4-방향 검색윈도우를 제안하며, Ⅲ장에서는 4-방향 검색을 지원하는 정수화소 움직임추정기(IME : Integer Motion Estimator) 아키텍처에 대하여 설명하고, Ⅳ장에서는 실험 및 결과에 대하여 정리하였으며, 마지막에서 고찰과 결론을 맺는다.

Ⅱ. 메모리 대역폭 개선

1-방향 검색윈도우는 그림 1(a)과 같이 스캔라인이 위에서 아래의 단일 방향으로 검색윈도우를 이동하여 래스터 방식으로 검색한다. 스캔경로상의 첫 번째 검색윈도우 ①을 처리하기 위하여 프레임 메모리로 부터 N×N의 데이터를 읽어오기 위한 초기 지연시간을 필요로 한다. 이후 검색윈도우를 ②~③에서 아래쪽으로 이동하면서 N×(N-1) 데이터를 재사용하고, N×1의 데이터를 새로 읽어 온다. 스캔경로가 "A"에서 "B"로 바뀔 때는 검색윈도우 ④에서 사용하였던 데이터를 재사용하지 못하고 검색윈도우 ⑤를 위한 데이터를 검색윈도우 ①에서와 같이 새롭게 읽어야 하여 스캔경로마다 초기 지연시간이 발생한다. 이같은 문제를 해결하고자 그림 1(b)의 3-방향 검색윈도우는 스캔경로가 "A"에서 "B"로 바뀔 때 1-방향 검색윈도우와 달리 수평방향으로 인접한 검색윈도우로 이동함으로써, 스캔경로마다 초기 지연시간을 필요로 하지 않고, 이전 스캔경로의 마지막 검색윈도우 ④의 데이터 중 (N-1)×N 데이터는 재사용하고 1×N 데이터를 추가로 읽어 와서 검색윈도우 ⑤를 구성한다. 그러나 이 방법도 다른 스캔경로의 검색윈도우 ②와 ⑦간 및 검색윈도우 ③과 ⑥간의 데이터 재사용을 처리하지 못하는 한계를 보이고 있다. 이와 같은 서로 다른 스캔경로상 검색윈도우간 데이터의 재사용을 위하여 스캔경로를 직선으로 스케쥴링하는 것이 아닌, 사행 형태의 스케쥴링을 적용할 수 있다. 사행 형태의 스캐닝을 진행하면, 인접경로에 위치한 검색윈도우 ②, ⑦과 검색윈도우 ③, ⑥간 각각 (N-1)×N의 데이터를 재사용할 수 있다. 스캔경로 방향이 아래에서 위로 바뀌는 검색윈도우 ④, ⑤와 같이 스캐닝을 진행하면 (N-1)×N의 데이터를 재사용할 수 있다. 이같은 2차원적 스캐닝은 스캔경로가 아래쪽(위쪽)으로 향할 때는 검색윈도우를 오른쪽, 오른쪽, 아래쪽, 왼쪽, 왼쪽, 아래쪽 (혹은 오른쪽, 오른쪽, 위쪽, 왼쪽, 왼쪽, 위쪽) 방향으로 사행 이동하는 스케쥴링 방법을 그림 1(c)에서 보이고 있다.

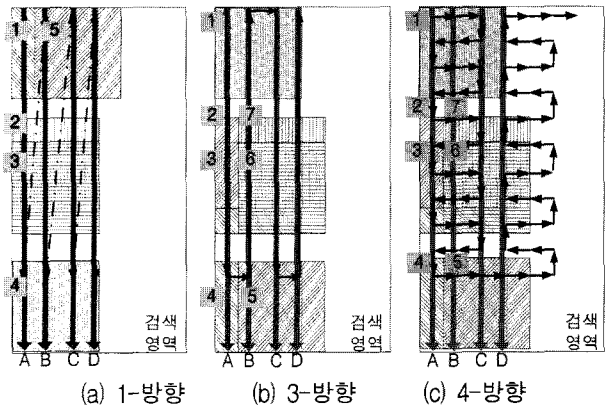


그림 1. 1/3/4-방향 검색윈도우
Fig. 1. 1/3/4-way search window.

본 연구에서는 스캔경로의 시작과 끝 부분의 데이터 재사용뿐만 아니라, 스캔경로의 중간부분에서 인접한 스캔경로와 중복되는 데이터를 재사용할 수 있는 사행 형태의 2차원 검색윈도우 스케쥴링인 4-방향 검색윈도우를 제안한다. 검색 영역 M×M과 매크로 블록 N×N에서 검색윈도우의 종류(1/3/4-방향)와 인접 스캔경로의 범위(ASR : Adjacent Scan-path Range)에 따라서 움직임추정에 요구되는 데이터 검색량을 분석하였다. 검색윈도우의 방법(1/3/4 방향)에 따른 메모리 대역폭은 검색영역(M×M)과 매크로블록(N×N)에 대해서 다음과 같이 각각 구해진다. 4방향 검색윈도우에서 ASR의 범위는 식(3)에 의해 1≤ASR≤M-(N-1)으로 제한된다.

$$N \times M \times (M-N+1) \quad \text{...(1) /*1-방향*/}$$

$$(N \times M) + (N \times (M-N+1)) \times (M-N) \quad \text{...(2) /*3-방향*/}$$

$$(N+ASR-1) \times M + \lceil (M-(N+ASR-1))/ASR \rceil \times ((N+ASR-1) \times (M-N) + (ASR+N)) \quad \text{...(3) /*4-방향*/}$$

이같은 메모리 대역폭은 검색영역의 M^2데이터 량에 비해서 검색윈도우 방법에 따라서 중복 검색의 정도가 차이가 있는 것을 보인다. 실제로 (M=48, N=16)일 때 표 1은 H.264의 데이터 압축률을 보장할 수 있는 규격의 메모리대역폭, 중복검색 빈도 및 검색개선 비율값을

표 1. 1/3/4 검색윈도우의 데이터 검색 처리규격 및 개선

Table 1. Data retrieval specification & enhancement of 1/3/4-way search window.

검색 윈도우	스캔 경로수	대역폭 (Bytes/MB)	중복검색 빈도	데이터 검색 개선율 (0/1/3/4-3/4-11)
0-방향	33	278,784	121	1/ 11/ 15.7/ 39.3/ 85.4
1-방향	33	25,344	11	0.091/ 1/ 1.44/ 3.5/ 7.8
3-방향	33	17,664	7.7	0.063/ 0.7/ 1/ 2.5/ 5.4
4-방향 (ASR=3)	11	7,104	3.08	0.025/ 0.28/ 0.4/ 1/ 2.2
4-방향 (ASR=11)	3	3,264	1.41	0.01/ 0.13/ 0.2/ 0.5/1

주)M=48, N=16

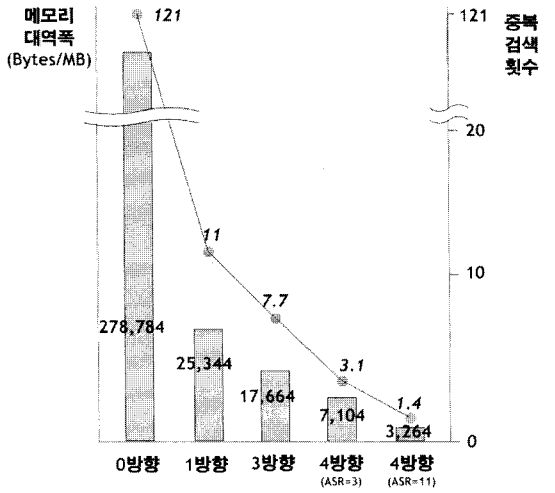


그림 2. 0/1/3/4-방향 검색위도우의 메모리 대역폭 및 중복검색횟수 비교

Fig. 2. Comparison of memory bandwidth and data Retrieval frequency for 0/1/3/4-way search window.

데이터 재사용이 없는 경우(0-방향)와 있는 경우(1/3/4-방향 ASR=3,11)에 대해서 표 1에서 비교하고 있다. 실제로 ASR=3의 4-방향 검색윈도우는 검색 빈도를 3.08회로 줄여서 2차원적 11회의 스캔경로에서 단지 7,104Bytes의 데이터 검색을 필요로 한다. 이는 중복 데이터 재사용이 없을 때보다 1/40로 메모리 대역폭의 검색을 개선하였으며, 1-방향 검색윈도우보다 1/3.5로 3-방향 검색윈도우보다 1/2.5의 메모리 대역폭 개선효과를 각각 보이고 있다. 이같은 0/1/3/4-방향 검색윈도우들에 대해서 메모리 대역폭 및 중복검색성능 차이를 비교해서 그림 2에 보이고 있다.

4-방향 검색윈도우는 인접한 윈도우 및 스캔경로들에 대한 2차원적인 데이터 재사용을 가능하게 해서 메모리 대역폭을 감소시킨다. 또한 인접한 하나의 스캔경로 뿐만 아니라 여러 개의 인접 스캔경로에 대한 데이터 재사용의 검색윈도우 설계에 고려하는 것이 가능한데, 포함되는 ASR(≥1)에 따라서 부가적으로 내부저장(쉬프트레지스터)이 필요하다. 4-방향 검색윈도우에서는 그림 1의 (c)에서와 같이 (ASR-1) 만큼 인접경로로 사행형태의 스캔하므로 인접경로 재사용을 위한 (ASR-1)×N크기의 레지스터어레이가 필요하며, 방향전환 시 ASR만큼 이동해야하므로 ASR×N크기의 레지스터어레이가 필요하다. 식(4)는 4-방향 검색 윈도우에서 ASR 크기에 변화에 따른 레지스터어레이의 크기(Reg)을 나타낸다.

$$\begin{aligned} Reg &= N\times(ASR-1) \quad ; \quad \text{인접경로재사용} \\ &\quad + N\times ASR \quad ; \quad \text{방향전환재사용} \\ &= N\times(2\times ASR-1) \end{aligned} \tag{4}$$

실제로 M=48, N=16일 때, 레지스터 어레이의 크기를 구해보면 식 (4)에서와 같이 16×2×(ASR-1)이다. 이와 같은 결과는 그림 3에서와 같이 ASR의 증가함에 따라 레지스터 어레이의 크기가 32Bytes씩 비례적으로 증가하며, 메모리 대역폭은 식 (3)과 같이 반비례적으로 감소함을 보인다.

III. 4방향 검색윈도우 지원 정수화소 움직임추정기 아키텍처

4-방향 검색윈도우를 위한 고속 H.264 정수화소 움직임추정(그림 4(a))은 2D PE어레이, RE어레이 그리고 SAD연산기로 구성된다. 2D PE어레이는 현재픽셀와 참조픽셀의 저장 및 연산에 1:1 대응되는 N×N개의 병렬 pe셀을 가지며, 스캔이동시 요구되는 많은 참조데이터

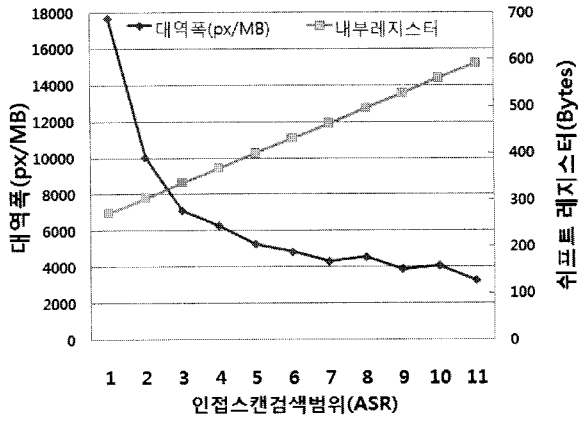
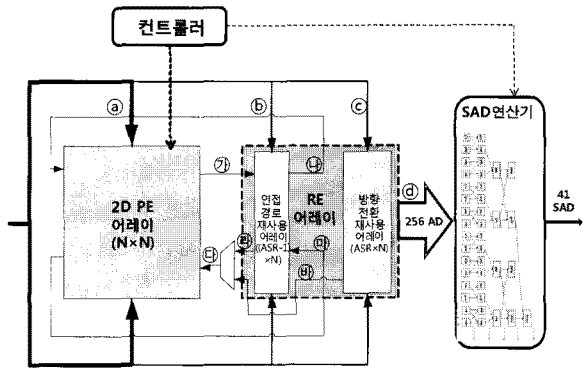


그림 3. 메모리 대역폭/레지스터크기와 ASR의 관계

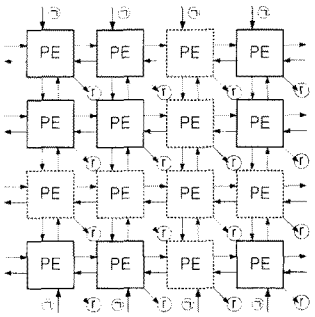
Fig. 3. Relation between memory bandwidth/register size and ASR.

의 입력을 인접한 pe셀 간에 참조데이터 이동으로 처리 가능하도록 위, 아래, 왼, 오른쪽으로 연결된 그림 4(b)의 메쉬형태 연결네트워크를 구성하였다. 그림 4(e)의 pe셀은 입력받은 참조화소를 멀티플렉서로 선별하여 내부 쉬프트레지스터에 저장하고 현재데이터와의 절대차분(Absolute Difference; AD)값 ⓡ을 출력하는 구조를 갖는다. RE어레이는 참조 픽셀 데이터의 저장 및 재사용을 위한 레지스터 블록이며, 인접경로 RE어레이와 방향전환 RE어레이로 구성된다. 인접경로 RE어레이는 2D PE어레이와 함께 (ASR-1)×N의 인접 스캔경로 참조데이터를 추가로 저장해서 공유할 수 있도록, 각각의 인접경로 re셀(그림 4(f))은 멀티플렉서로 참조데이터를 선별하여 쉬프트레지스터에 저장하고 그림 4(c)와 같은 메쉬 네트워크를 구성한다. 방향전환어레이는 스캔 진행방향의 끝지점에서 ASR×N의 인접스캔경로 데이터를 추가로 저장해서 스캔방향이 역전될 때 연속적인 2차원적 스캔경로를 위하여 인접한 경로의 참조데이터를 공유할 수 있도록 방향전환 re셀(그림 4(g))을 위, 아래, 왼쪽으로 연결한 그림 4(d)의 네트워크를 구성한다. 방향전환의 검색은 항상 오른쪽으로 진행하기 때문에 오른쪽에 저장된 참조데이터를 공유하기 위하여 왼쪽으로 참조데이터를 쉬프트하는 레지스터 네트워크를 구성한다.

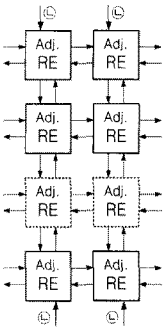
실제로 정수화소 움직임추정기는 "초기화"이후 "사행검색", "방향전환준비" 및 "스캔방향전환" 과정의 동작을 그림 5와 같이 반복하게 된다. 초기화는 매크로블록의 현재데이터를 외부버퍼로 부터 2D PE어레이로 입력시키며, 참조데이터는 참조메모리로 부터 2D PE어레이와 RE어레이로 입력된다. 2D PE어레이에는 움직임추정의 대상인 현재데이터와 검색영역의 참조데이터가 저장되는데, 이는 초기지연시간(latency) 동안에 외부버퍼로 부터 각각 현재데이터 N개의 픽셀 및 참조데이터 (N+ASR-1)픽셀 단위로 N사이클 동안 그림 4(a)의 ㉠과 ㉡버스를 통해서 2D PE어레이와 인접경로 RE어레이에 입력된다. 이같은 "초기화" 과정에 따라서 2D PE어레이는 검색영역내의 최초 검색점 ⓐ에 대한 N×N의 현재데이터와 참조데이터의 차이값(AD)을 처리하고, N2개의 AD를 병렬 출력한다. ASR=3일때 ⓐ에서 ⓑ, ⓑ에서 ⓒ같은 그림 4(a)의 검색점의 오른쪽이동에 대해서 2D PE어레이의 우측부분으로 인접경로 RE어레이에 저장된 인접한 경로의 1×N개의 참조 데이터를 ㉮버스에서 ㉯버스로 입력받으며, 2D PE어레이와 인접경로



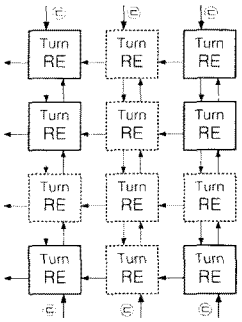
(a) 정수화소 움직임추정기 블록도



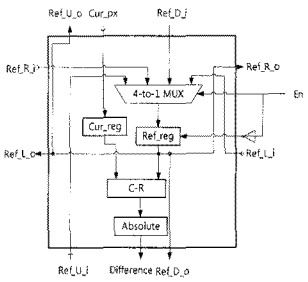
(b) 2D PE 어레이 네트워크



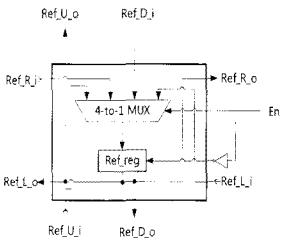
(c) 인접경로 RE어레이 네트워크



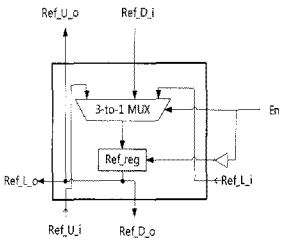
(d) 방향전환 RE어레이 네트워크



(e) pe셀구조



(f) 인접경로 re셀구조



(g) 방향전환 re셀구조

그림 4. 정수화소 움직임추정 아키텍처
Fig. 4. Integer Motion Estimation Architecture.

RE어레이의 참조데이터는 모두 좌측으로 이동하고 2D PE어레이의 가장 왼쪽 열의 데이터는 ㉮버스를 통해 인접경로 RE어레이로 순환(Cyclic) 저장된다. 이 과정에서 검색윈도우 A는 (ASR-1)만큼 쉬프트해서 검색윈도우 B까지 진행하며, 검색점마다 N2개의 AD들이 병렬연산되어 출력된다.

그림 5의 검색점 ⓒ에서 ⓓ같이 아래로 이동하는 경우 외부버퍼로부터의 N개와 (ASR-1)개의 참조데이터가 각각 아래의 ㉠ 및 ㉡버스로 2D PE어레이와 인접경로 RE어레이에 병렬 입력되며 동시에 2D PE어레이와 인접경로 RE어레이의 모든 데이터들은 위쪽으로 쉬프트 저장된다. 검색점 ⓓ에서 ⓔ, 검색점 ⓔ에서 ⓕ같이 왼쪽으로 이동하면 2D PE어레이의 좌측부분으로 인접경로 RE어레이의 저장된 인접한 경로의 1×N개의 참조 데이터를 ㉯버스로 입력받으며, 2D PE어레이와 인접경로 RE어레이의 참조데이터는 함께 우측으로 이동하고 2D PE어레이의 가장 오른쪽 열의 1×N개의 데이터는 ㉰버스를 통해 인접경로 RE어레이로 저장된다. 이때에도 매 검색점마다 (N-1)×N개의 참조데이터가 재사용되면서 N2개의 AD가 병렬 연산된다. 오른쪽→오른쪽→아래쪽→왼쪽→왼쪽의 2차원적인 "사행검색" 과정은 그림 5의 검색점 ⓖ까지 반복되고 이후 그림 4의 정수화소 움직임추정기는 "방향전환준비" 과정을 시작하게 된다.

검색점 ⓗ로 아래쪽 검색이 진행될 때 (N+ASR-1) 참조데이터는 ㉠, ㉡ 버스를 통해서 2D PE 어레이와 인접경로 RE어레이로 입력되며, 동시에 검색윈도우 D에 인접된 ASR개의 참조데이터가 ㉢버스로 방향전환 RE어레이로 병렬 입력되기 시작한다. 이와 같은 "방향전환준비" 과정에서는 (N+2ASR-1)참조데이터의 입력을 검색점 ⓘ까지 진행하면서 검색윈도우 E까지 참조데이터를 재사용하는 것이 가능하도록 한다. 즉 방향전환 RE어레이에 ASR×N의 참조데이터를 저장해서, 검색점 ⓘ에서 ⓜ까지의 "스캔방향전환" 과정은 참조데이터를 연속적으로 재사용 가능하도록 한다.

실제로 스캔방향을 전환(아래쪽→위쪽)할 때인 검색점 ⓘ(검색윈도우 C)에서 검색점 ⓙ(검색윈도우 D)로 이동하는 동안 검색점마다 우측부분으로 인접경로 RE어레이의 저장된 인접한 경로의 1×N개의 참조 데이터를 ㉱버스에서 ㉲버스로 입력받는다. 2D PE어레이내의 참조데이터는 모두 좌측으로 이동하고 2D PE어레이의 가장 왼쪽 열의 데이터는 ㉮버스를 통해 인접경로 RE

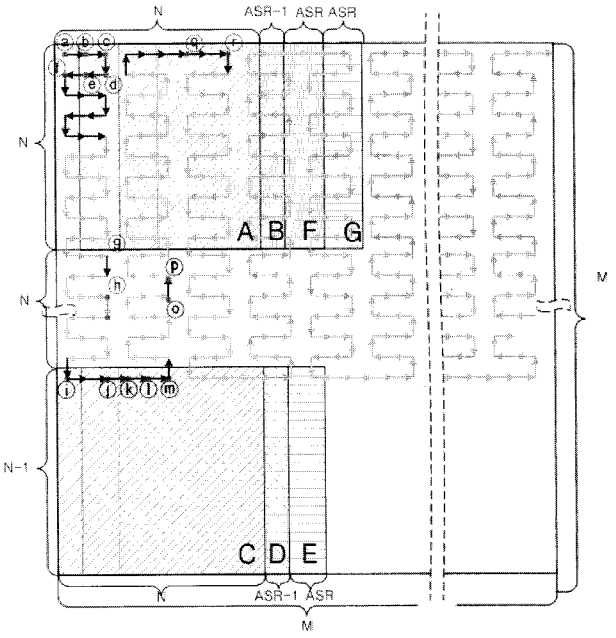


그림 5. 4-방향 검색윈도우 스캔과정
Fig. 5. 4-way(ASR=3) search window scanning procedures.

어레이로 순환 저장되며 (N-1)×N개의 참조데이터가 재사용된다. 방향전환을 위해서 검색점 ⓚ에서 검색점 ⓛ로 이동하는 과정부터 ⓗ에서 ⓘ까지의 "방향전환준비" 과정에서 방향전환 RE어레이에 저장된 검색윈도우 E까지의 참조데이터를 사용하게 된다. 방향전환 RE어레이 참조데이터는 ㉰버스에서 ㉲버스를 통하여 2D PE어레이 우측으로 1×N개의 데이터가 입력되고, 2D PE어레이 좌측에서는 1×N개의 데이터가 ㉮버스를 통하여 인접경로 RE어레이에 저장된다. 이와 같이 검색점 ⓚ부터 검색점 ⓜ까지 과정은 방향전환 RE어레이의 데이터를 재사용하면서 검색윈도우 E까지 진행한다. 검색점 ⓜ이후부터 검색진행방향이 아래쪽에서 위쪽으로 바뀌고 2차원적 검색경로폭이 ASR만큼 이동하면서 "사행검색" 과정에서 "스캔방향전환" 과정까지의 동작을 반복한다.

SAD연산은 2D PE어레이에서 출력된 256AD값을 가변블록 계산에 요구되는 4×4부터 16×16까지 41가지 합산해서 SAD값들을 구한다. 실제로 41가지 SAD연산값들은 서로 입력의 범위가 다르기 때문에 다수 입력 및 연산을 공유하는 그림 6과 같은 트리구조의 16×16 전체 SAD 연산과정이 적합하다. 이와 같은 SAD연산과정은 전체임계경로가 큰문제를 갖는데, 전체 256 AD 연산결과 값이 5단의 덧셈을 통해서 구해지게 된다. 이 같은 한계는 그림 6의 연산과정을 다수 AD에 대한 덧

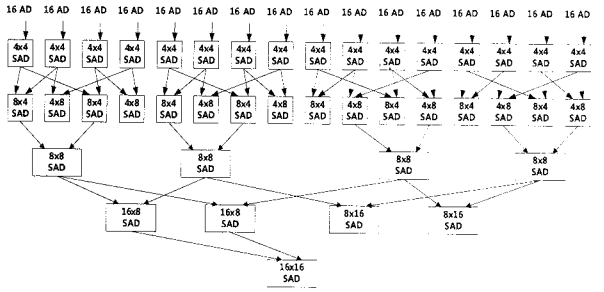


그림 6. 가변블록 SAD 연산 과정
Fig. 6. Variable block size SAD computing procedure.

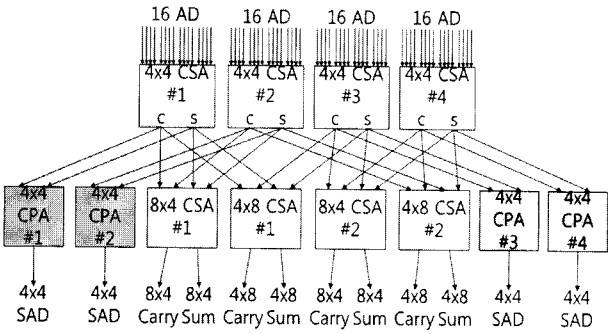
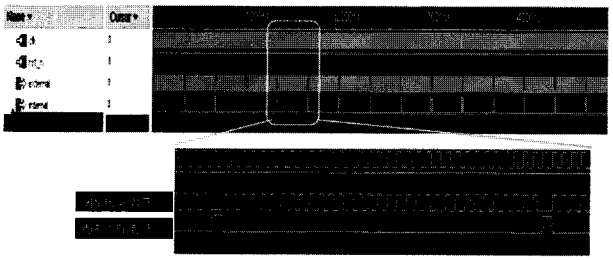


그림 7. 4×4, 4×8, 8×4 블록 SAD 연산 구조
Fig. 7. 4×4, 4×8, 8×4 block SAD computing architecture.

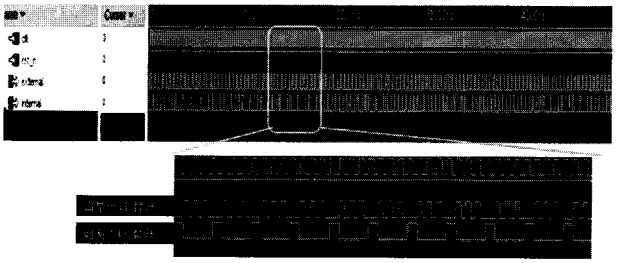
셈을 고속처리하는 CSA(Carry Save Adder)기반 Wallace트리 형태로 구현해서 해결할 수 있다. 4×4블록단위의 16 AD입력에 대해서 16×16블록 전체의 CSA Wallace트리를 구성하고, 정해진 가변블록단위 별로 SAD값을 계산하기 위해 트리의 노드값을 합산하는 덧셈회로를 병렬구성한다. 이때 트리의 중간노드로부터 블록단위 SAD값을 계산하는 과정은 16×16전체블록의 SAD 연산과정과 병행수행되기 때문에 연산속도보다는 크기가 작은 기본적인 CPA(Carry Propagation Adder)를 이용할 수 있다. 전체임계경로를 결정하는 256 SAD연산과정에서는 고속의 CLA(Carry Lookahead Adder)를 활용해서 전체 SAD연산회로의 성능을 높인다. 그림 7은 4×4 SAD연산과정이 효과적으로 병렬 구현됨을 보이고 있다.

Ⅳ. 실험 및 고찰

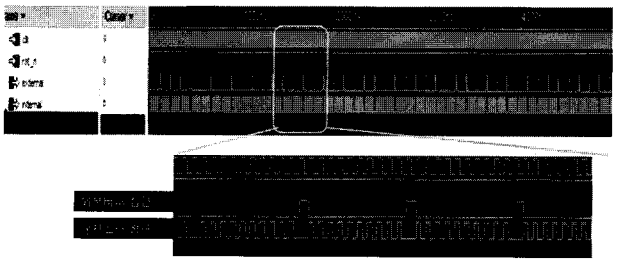
H.264 정수화소 움직임추정 아키텍처를 위해서 제안된 4-방향 검색윈도우의 설계검증 및 성능 비교 실험들이 진행되었다. 설계검증은 2D PE어레이 구조가 3-방향(ASR=1), 4-방향(ASR=3,11)의 세가지 16×16 검색윈도우들이 48×48검색영역의 움직임추정에 필요한 참조데이터 검색을 정량적으로 비교측정 하였다. 접근고속화를 검증하기 위하여 검색윈도우 별 외부버퍼 와 내부레지스터 접근 횟수를 시뮬레이션 하였다. 또한 참조데이터 입력량의 감소를 확인하기 위해서 인접스캔경로 ASR=1 (3-방향), 3과 11(4-방향)의 세 가지 16×16 검색윈도우들이 48×48검색영역의 움직임추정에 필요한 참조데이터 검색을 정량적으로 비교 측정하였다. 마지막으로 구현된 정수화소 움직임추정 아키텍처의 동작성능을 확인하기 위해서 4-방향(ASR=3)의 검색윈도우를 Magnachip 0.18um 공정을 통해서 비교/검증 하였다. 정수화소 움직임추정 아키텍처에서 2D PE어레이의 검색윈도우 스케줄링 방법은 참조데이터를 2D PE어레이의 외부버퍼로부터 공급받거나 어레이 내부의 재사용



(a) ASR : 1(3-방향)



(b) ASR : 3(4-방향)



(c) ASR : 11(4-방향)

그림 8. ASR=1(3-방향)과 ASR=3/11(4-방향) 검색윈도우의 외부버퍼 및 내부재사용레지스터에 대한 검색사이클
Fig. 8. Retrieval cycles for reuse external buffer and internal register of ASR=1(3-way) and ASR=3/11(4-way)

표 2. ASR=1(3-방향)과 ASR=3/11(4-방향)의 입력재사용 검색사이클 비교

Table 2. Comparison for input and reuse retrieval cycles of ASR=1(3-way) and ASR=3/11(4-way).

ASR	입력 사이클		재사용 사이클	총 검색 사이클
	초기대기사이클	스캔 사이클		
1	1072		32	1104
	16	1056		
3	369		735	1104
	16	353		
11	112		992	1104
	16	96		

(검색영역 : 48×48)

레지스터로부터 입력받는 정도를 결정하게 된다. 4방향(ASR=3,11) 검색윈도우에 대한 외부버퍼와 재사용 레지스터부터의 참조데이터 검색사이클을 비교해서 보이고 있다. ASR값이 커짐에 따라서 외부버퍼의 검색은 감소되고 내부재사용 레지스터에 대한 검색은 증가되고 있음을 확인할 수 있다.

실제로 48×48 검색영역에 대하여 2D PE어레이의 16×16검색윈도우종류에 따른 입력과 재사용의 차이를 외부버퍼와 내부레지스터의 검색 사이클로써 비교한 결과가 표 2에 정량적으로 비교되어 있다. 전체 검색 사이클은 1104로 동일하지만 2차원 검색범위 즉 ASR이 커짐에 따라서 재사용 사이클이 증가됨을 보이고 있다. 여기 내부레지스터가 외부버퍼보다 구조적으로 높은 메모리 대역폭을 제공할 수 있다는 점을 고려한다면, 4방향 ASR=11의 검색윈도우가 참조데이터 검색에 가장 높은 성능을 제공하게 된다.

그림 4(a)의 정수화소 움직임추정기의 데이터패스는 AD계산의 2D PE어레이와 41가지 덧셈을 구하는 SAD 연산기로 구성된 임계경로를 갖는다. 그림 9은 전체임계경로를 Magnachip 0.18um에서 세부블록별로 구분해서 보이는데, SAD 연산과정을 4×4 기본블록의 16 SAD연산시간부터 16×16전체블록의 256 SAD연산시간까지의 임계경로차이를 나타내고 있다. 정수화소 움직임 추정기의 전체 데이터 패스를 2단 파이프라인으로 설계할 때, 2D PE어레이부터 4×8 및 8×4의 SAD연산까지의 임계지연시간을 파이프라인 스테이지 단위로 하는 2단 파이프라인을 구성해서 서치포인트당 6.259ns의 연산성능을 Magnachip 0.18um에서 구현할 수 있음을 보인다.

2D PE 4-방향 검색윈도우 성능검증을 위해 CIF,

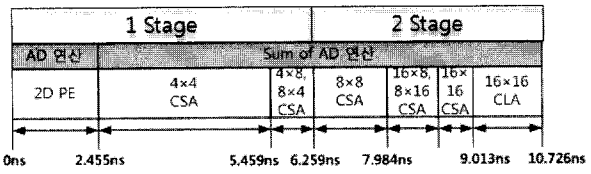


그림 9. 정수화소 움직임추정의 임계경로

Fig. 9. Critical path in integer motion estimation.

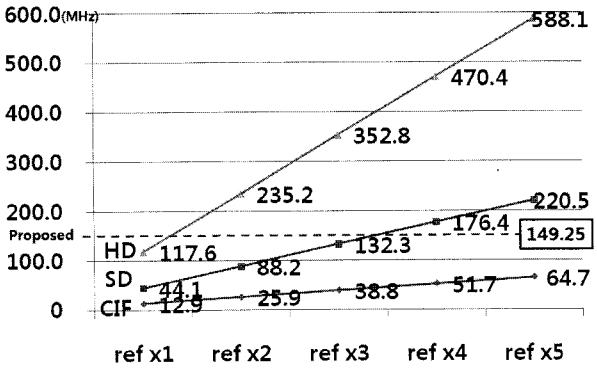


그림 10. 4-방향 검색윈도우의 처리성능

Fig. 10. Performance for 4-way search window.

표 3. 기존 움직임추정 성능비교

Table 3. Performance comparison for Motion Estimation.

Architecture	[2]	[4]	[5]	[8]	본연구
Search range	32×32 (1/2)	64×64 (2)	64×32 (1)	48×32 (1/2)	48×48 (1)
Reference frame	1(1)	1(1)	1(1)	1(1)	1(1)
Resolution	720×576 (1/2)	704×576 (1/2)	1280×720 (1)	720×480 (1/2)	1280×720 (1)
Frame rate(fps)	30(1)	15(1/2)	30(1)	30(1)	30(1)
본 연구 대비 총 연산량	1/4	1/2	1	1/4	1
Process (μm)	0.25um	TSMC .18um	UMC 0.18um	0.35um	0.18um
Logic (gates)	105K	154K	330.2K	106K	214K
On-chip memory(Kbits)	0	60	208	25	0
Working frequency(MHz)	52	100	108	64.11	149.25

SD, HD영상포맷에 대해 실시간처리 주파수에 따른 동작주파수를 비교하였다(그림 10). 4-방향 검색윈도우는 CIF영상은 참조프레임 5장, SD영상은 참조프레임 3장 이상도 처리가 가능하며 HD영상에 대해 참조프레임 1장을 실시간 처리가능하다.

표 3은 정수화소 움직임 추정기(IME)를 위한 기존연구[2, 4~5, 8]과 본연구의 처리 성능 및 사양을 비교해서 보인다. Search range와 reference frame, Resolution 및 Frame rate(fps)등은 IME가 처리해야하는 연산량과 직

접적인 관계를 갖는데, 본 연구를 기준으로 다른 연구의 연산량 비율을 ()에 나타내고 있다. 실제로 [2]/[4]/[5]/[8]의 IME 구조들은 각각 (1/4)/(1/2)/(1)/(1/4)의 연산량을 처리하기 위하여 0.18~0.35um의 공정을 통해서 Logic 및 On-chip memory를 설계 구현하여 Working Frequency에서 동작시키게 된다. 본 연구는 연산량 비율 〈1의 다른 연구들에 비교해서 2DPE로써 4방향 pe셀 구조 때문에 다소 큰 구현 비용을 부담해야 함을 보인다. 하지만 대등한 연산 처리량을 갖는 연구 [5]에 비해서는 2/3의 Logic 구현면적으로 메모리 구현 부담이 없는 결과를 보이고 있다. 또한 On-chip메모리가 없기 때문에 2D PE의 요구대역폭이 높아서 [5]연구보다 1.5배 높은 동작주파수를 요구함을 나타내고 있다.

V. 결 론

본 연구에서는 고속 H.264 정수화소 움직임추정의 요구하는 메모리 대역폭을 감소시키는 4-방향 검색윈도우와 병렬 연산구조를 제안하고 설계하였다. 2D PE 어레이구조와 병렬연산에 요구되는 메모리대역폭을 최소화하여 HD급 영상을 실시간 처리하도록 구현하였다. 움직임추정 아키텍처는 2D PE어레이에 RE어레이를 추가해 2차원적으로 인접경로간의 참조데이터를 재사용하여 메모리 대역폭을 3-방향 검색윈도우보다 40%로 개선시켰다. 움직임추정의 임계경로의 대부분을 차지하는 SAD연산의 가속을 위하여 CSA덧셈기를 적용하고 이를 2단 파이프라인 구성하여 성능을 향상시켰다. 제안된 4-방향 검색 정수화소 움직임추정구조는 Magnachip 0.18um공정으로 214K게이트로 구현해서 동작속도 149.25MHz에서 동작함을 확인하였다. 기존 연구들에 비해 4-방향 검색 정수화소 움직임추정기는 작은 면적을 갖고 메모리 대역폭을 감소시켜서 48×48 검색영역과 검색윈도우 16×16로써 참조프레임 1장의 HD급 동영상을 30프레임 실시간 처리함을 확인하였다.

참 고 문 헌

[1] Yang Song, Zhenyu LIU, "VLSI Architecture for Variable Block Size Motion Estimation in H.264/AVC with low Cost Memory organization", IEEE, VLSI Design, Automation and Test, 2006 International Symposium, pp. 1-4, April 2006.

[2] Xiang Li, Rahul Chopa, Kenneth W. Hsu, "Novel VLSI Architecture of Motion Estimation for H.264 standard", IEEE, SOC Conference, 2005 Proceedings. IEEE International, pp. 117-118, Sept 2005.

[3] Kenneth W. Hsu, Xiang Li, Rahul Chopa, "An IC Design for Real-Time Motion Estimation in H.264 Digital Video", IEEE, Circuits and Systems, 2005 48th Midwest Symposium, vol 2, pp. 1489-1493, Aug 2005.

[4] Minho Kim, Ingu Hwang, Soo-Ik Chae, "A Fast VLSI Architecture for Full-Search Variable Block Size Motion Estimation in MPEG-4 AVC/H.264", IEEE, Design Automation Conference, 2005 Proceedings of the ASP-DAC 2005 Asia and South Pacific, vol 1, pp. 631-634, Jan. 2005.

[5] Dong-Xiao Li, Wei Zheng, Ming Zhang, "Architecture Design for H.264/AVC Integer Motion Estimation with Minimum Memory Bandwidth", IEEE Transactions on Consumer Electronics, Vol. 53, No. 3, AUGUST 2007

[6] Swee Yeow Yap, John V. McCanny, "A VLSI Architecture for Variable Block Size Video Motion Estimation", IEEE, Circuits and Systems II: Express Briefs, IEEE Transactions, vol 51, pp. 384-389, July 2004.

[7] Swee Yeow Yap, John V. McCanny, "A VLSI Architecture for Advanced Video Coding Motion Estimation", IEEE, Application-SPEcific Systems, Architectures, and Processors, 2003 Proceedings. IEEE International Conference, pp. 293-301, June 2003.

[8] Yu- Wen Huang, Tu-Chih Wang, Bing-Yu Hsieh, and Liang-Gee Chen, "Hardware Architecture Design for Variable Block Size Motion Estimation in MPEG-4 AVC/JVT/ITU-T H.264", IEEE, Circuits and Systems, 2003 ISCAS '03 Proceedings of the 2003 International Symposium, vol 2, pp. 796, 799, May 2003.

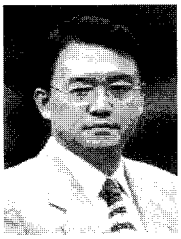
[9] Ching-Yeh Chen, Shao-Yi Chien, Yu-Wen Huang, Tung-Chien Chen, Tu-Chih Wang, and Liang-Gee Chen,, "Analysis and Architecture Design of Variable Block-Size Motion Estimation for H.264/AVC", Circuits and Systems I: Regular Papers, IEEE Transactions on, vol 53, pp. 578-593, March 2006.

[10] Zhenyu Liu, Yang Song, Ming Shao, Shen Li, Lingfeng Li, Satoshi Goto and Takeshi Ikenaga, "32-Parallel SAD Tree Hardwired Engine for Variable Block Size Motion Estimation in

HDTV1080P Real-Time Encoding Application", Signal Processing Systems, 2007 IEEE Workshop on, pp. 675-680, Oct. 2007.

저 자 소 개

고 병 수(정회원)
2003년 광운대학교 컴퓨터공학과 학사 졸업.
2005년 광운대학교 컴퓨터공학과 석사 졸업.
2005년～현재 광운대학교 컴퓨터공학과 박사과정
<주관심분야 : 영상신호처리, SoC설계, 임베디드 시스템>



공 진 흥(평생회원)
1980년 서울대학교 전자공학과 학사 졸업.
1982년 한국과학기술원 전기 및 전자공학과 석사 졸업.
1989년 텍사스주립대학교 컴퓨터공학과 박사 졸업.
<주관심분야 : 영상신호처리, SoC설계, 임베디드 시스템>