논문 2011-48SD-12-9

고속 탐색 알고리즘에 적합한 움직임 추정 전용 명령어 및 구조 설계

(Novel IME Instructions and their Hardware Architecture for Fast Search Algorithm)

방 호 일*, 선우 명 훈**

(Ho Il Bang and Myung Hoon Sunwoo)

요 약

본 논문은 H.264/AVC, MPEG4 등, 다양한 영상압축 코덱을 지원할 수 있는 ME ASIP (Application-specific Instruction Processor)의 정화소 움직임 추정 전용 명령어와 재구성 가능한 하드웨어 구조를 제안한다. 제안하는 전용의 명령어와 하드웨어 가속기는 HD급의 고화질 영상을 지원할 수 있는 성능을 가지고 있다. 제안하는 정화소 움직임 추정 명령어는 다수의 병렬 연산과 패턴 정보를 이용한 가변 포인트 2D SAD 연산기 구조를 통하여 전역탐색을 비롯한 각종 고속 탐색 알고리즘을 지원한다. 이를 위한 하드웨어 구조는 128개의 Processor Elements (PEs)로 구성되어 있는 Processor Element Group (PEG) 하나당 25,500 게이트를 가진다. 제안하는 ASIP은 Synopsys 사의 Processor Designer 로 검증하였고, Design Compiler를 이용 IBM 90nm 공정으로 합성하였다. 그 결과 제안하는 ASIP의 하드웨어 사이즈는 453K 게이트였으며, 동작 주파수는 188MHz로 HD급 1080p의 해상도를 가지는 영상을 실시간으로 동작 시킬 수 있다. 본 논문은 기존 2D SAD ASIP에 비하여 하드웨어 사이즈 측면에서 26%, 연산 속도 측면에서 평균 18%의 성능 향상을 보인다.

Abstract

This paper presents an ASIP (Application-specific Instruction Processor) for motion estimation that employs specific IME instructions and its programmable and reconfigurable hardware architecture for various video codecs, such as H.264/AVC, MPEG4, etc. With the proposed specific instructions and variable point 2D SAD hardware accelerator, it can handle the real-time processing requirement of High Definition (HD) video. With the SAD unit and its parallel operations using pattern information, the proposed IME instructions support not only full search algorithms but also other fast search algorithms. The hardware size is 25.5K gates for each Processing Element Group (PEG) which has 128 SAD Processor Elements (PEs). The proposed ASIP has been verified by the Synopsys Processor Designer and implemented by the Design Compiler using the IBM 90nm process technology. The hardware size is 453K gates for the IME unit and the operating frequency is 188MHz for 1080p@30 frame in real time. The proposed ASIP can reduce the hardware size about 26% and the number of operation cycles about 18%.

Keywords : H.264/AVC, video coding, ASIP, reconfigurable coding, motion estimation

* 학생회원, ** 정회원-교신저자, 아주대학교 정보통신대학 전자공학부 SoC 연구실
(School of Electrical and Computer Engineering, Ajou University, SoC LAB)

※ This work was supported by Mid-career Researcher Program through the NRF grant funded by the MEST (20110016671), by the framework of international cooperation program managed by National Research Foundation of Korea(2011-0030930) and by IC Design Education Center (IDEC).
접수일자: 2011년7월25일, 수정완료일: 2011년12월2일

Ⅰ. 서 론

DTV, 휴대폰 등 다양한 디지털 멀티미디어 기기들의 발전으로 고화질 멀티미디어 영상에 대한 관심과 수요가 크게 증가하고 있다. 특히 H.264/AVC[1]는 Variable Block Matching 기술과 다양한 정화소 및 부화소 움직임 추정 방법을 이용하여 기존 영상 압축 표준에 비해 높은 효율의 성능을 보여준다[2][3].

영상 압축 표준은 적용 대상이나 플랫폼에 따라 소프트웨어와 하드웨어로 구현된다. 특히 고해상도 실시간 처리를 위해서는 하드웨어 구현이 필수적이다. 현재까지는 Application Specific Integration Circuit (ASIC) 방식의 하드웨어 구현이 주로 연구 되었다. 복잡한 연산기 구조 때문에 필요한 동작 성능을 만족하기 위해서는 특정 알고리즘에 최적화하여 ASIC으로 구현하는 것이 최선의 방식이었다. 하지만 ASIC으로 구현하는 것은 개발 시간과 비용이 많이 들고, 성능 향상을 위한 개선 작업이 어렵다는 단점이 있다.

ASIC의 단점을 개선하기 위해서 Application Specific Instruction-set Processor (ASIP) 방식의 구현 방식이 새로운 대안이 되고 있다. ASIP 방식은 특정 응용 분야에 대한 전용의 명령어 셋을 가지고 있는 프로세서를 개발하여, C 코드나 어셈블리 코드와 같은 프로그램 가능한 언어를 이용한 제어를 통해 효율적이고 유연한 형태로 구현이 가능하다[4].

본 논문에서는 효율적인 움직임 추정 연산이 가능하도록 움직임 추정 전용 ASIP을 위한 Sum of Absolute Different (SAD)를 제안한다. 특히 최근에 제안되고 있는 고속탐색 알고리즘들을 효율적으로 지원할 수 있는 명령어를 제안한다. 그리고 이들 명령어를 위한 하드웨어 구조를 보여준다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 DSP, Intel 의 MMX4.1 그리고 기존 ASIP구조에서의 움직임 추정의 구현에 대하여 설명하고 Ⅲ장에서는 새로운 구조에서의 정화소 움직임 추정 명령어를 제안한다. Ⅳ장에서는 제안하는 명령어의 하드웨어 구조에 대해서 설명한다. Ⅴ장 에서는 기존의 ASIP과 성능을 비교한다. 마지막으로 Ⅵ장에서 결론을 맺는다.

Ⅱ. 기존 프로세서 구조

2.1. 기존 DSP 구조

표 1은 일반적인 DSP명령어를 이용한 SAD 연산 수행 방식이다. 일반적인 명령어로 1 픽셀에 대한 SAD 연산을 수행하기 위해서는 DSP 명령어로 3 사이클의 명령어가 필요하다[5]. 최근의 1080p와 같은 높은 해상도를 처리하는데 위의 명령어를 사용하게 되면 연산 부하가 매우 커지게 된다. 때문에 Intel의 Streaming SIMD Extensions (SSE)에서는 Packed Sum of Absolute Differences Byte Word (PSADBW) 명령어를 지원하여 명령어당 64bit, 8 픽셀의 SAD 연산을 수행하였다[6]. 하지만 H.264와 같은 연산량이 많은 표준을 효과적으로 지원하기 위해 더 큰 처리량을 필요로 하였고, 이를 만족시키기 위해 SSE4.1에서는 그림 1과 같이 Multiple Packed Sums of Absolute Difference Byte Word (MPSADBW) 명령어를 제안하였다[7].

MPSADBW 명령은 PSADBW 대비 처리량이 4배로, 명령어당 8개의 4바이트 길이 SAD 연산을 수행한다. H.264의 기본 처리 단위인 16x16 매크로 블록에 대한 연산을 하기 위해서는 8개의 명령어를 필요로 하게 된다. 따라서, 64x64의 탐색창을 갖는 1080p의 HD급 영상을 전역 탐색으로 처리하기 위해서는 267,386,880 번의 연산을 필요로 하다. 이를 실시간으로 지원하기 위해서는 33.3ms안에 처리해야 함으로 프로세서가 8GHz이상으로 동작해야 한다. 정화소 움직임 추정에만 위의 연산을 수행하기 위해서는 소모되는 전력도 커질 뿐 아니라, 사이즈도 ASIC보다 크기 때문에 비효율적이다.

표 1. TMS320C6X DSP의 SAD 명령어
Table 1. SAD instruction of TMS230C6X.

동작

SUB R0, R1, R2
ABS R2
ADD R2, R3, R3

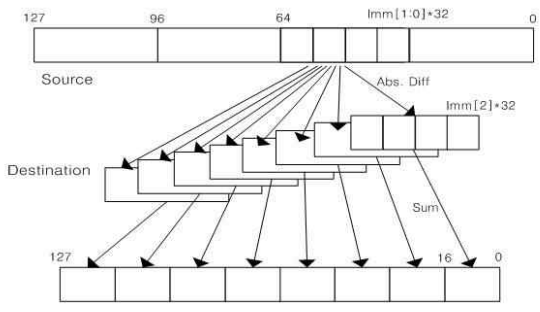

그림 1. SSE4.1의 MPSADBW 명령어
Fig. 1. MPSADBW operation of SSE4.1.

2.2. 기존 ASIP구조

앞서 연구된 ASIP[8]은 4x4블록에 대해서 한 번에 연산 할 수 있는 SAD 명령어를 제안하고 있다. 연산된 4x4블록 모드의 SAD 결과를 이용해서 소프트웨어적 처리를 통해 8x8, 16x16등 기타 모드들에 대한 모든 연산이 가능하게 된다. 따라서 16x16모드에 대한 결과를 얻기 위해서는 16번의 SAD 명령의 반복이 필요하다. 마찬가지로 (-32~+32) 탐색창의 1080p의 HD급 영상을 전역 탐색으로 처리하기 위해서는 534,773,760번의 연산을 필요로 하기 때문에 실시간 처리를 위해 약 16GHz의 동작 속도를 유지해야 한다. 실제로 이러한 속도에는 성능이 미치지 못하기 때문에 이 논문에서는 Enhanced Predictive Zonal Search (EPZS)에 최적화하여 설계하여 탐색 지점을 줄여 처리한다. 이 논문에서는 이러한 적응적 탐색의 경우 분기 명령어 등과 같은 프로그램 제어 유닛 명령어를 이용해서 적응적으로 탐색이 가능한 어셈블리 코드를 지원하도록 하였다.

또 다른 ASIP[9]에서는 16x16블록단위의 하나의 탐색 지점에 대한 연산을 수행하는 명령어를 제안하였다. 연산양이 많은 어플리케이션을 지원하기 위해서 병렬로 연산기를 두어, 보다 많은 연산을 빠르게 수행할 수 있도록 지원하고 있다. 하지만 이 논문에서는 연산양을 줄이기 위해서 VBM 알고리즘을 전부 지원하지 않고 16x16, 8x8모드에 대해서만 지원을 하도록 설계되었다. 그리고 앞서 언급한 대로 재구성 가능한 하드웨어를 지원하기 때문에 하드웨어 크기와 속도를 사용자가 조절할 수 있도록 제안하였다. 하지만 이 논문 역시 HD급 전역탐색의 경우 실시간 처리를 지원하기 어려운 단점이 있다. 때문에 [9]는 고속 탐색 중에서도 탐색 지점수가 적고 패턴이 단조로운 고속 탐색에 대해서만 지원하고 있다.

Ⅲ. 제안하는 정화소 움직임 추정 명령어 구조

본 절에서는 효율적으로 정화소 움직임 추정을 위한 ASIP 전용의 명령어를 제안한다. HD급의 높은 해상도를 처리하기 위해서 빠르고 많은 연산을 병렬 처리할 수 있는 구조가 필요하다. 그리고 다양한 고속 탐색 지원을 위해 명령어는 다양한 패턴을 지원하는 구조로 정의 되어야 한다.

본 절에서는 효율적으로 SAD 연산을 수행하는 명령어와 다양한 패턴을 지원하기 위한 패턴레지스터를 제안한다.

3.1. 다양한 패턴레지스터 타입

제안하는 Variable Block SAD (VBSAD) 명령어는 탐색 시작 지점과 패턴 정보, 연산 크기를 오퍼랜드로 입력 받아서 수행한다. 이에 따라 두 가지 형태의 패턴레지스터를 제안하였다. 첫 번째 패턴레지스터A는 마스크 방식의 레지스터로써 최대 8x16 지점에 대한 연산을 수행하는 VBSAD 명령어를 제어하기 위해 128비트 크기의 레지스터에 탐색을 원하는 지점을 1로 표현한다. 따라서 패턴레지스터A는 다이아몬드탐색, 헥사곤 탐색, 3SS와 같이 탐색 원점으로부터 가까운 거리의 정방형 탐색 패턴을 표현하는데 효과적이다.

그림 2에 표현된 패턴레지스터B는 최대크기 16x16 사이즈의 탐색영역에 대하여 탐색지점 수와 탐색 시작 지점으로부터 가로방향의 탐색 지점이 떨어진 거리 정보를 저장한다. 이때 최대 탐색 포인트는 가로 방향 8 포인트, 세로방향 16 포인트 그리고 최대 거리는 15로 두었다. 따라서 패턴레지스터B는 MDHC (Mixed Diamond, Hexagon and Cross) 탐색, UMH (Unsymmetrical-cross Multi-Hexagon-grid) 탐색, DVS (Dynamically Variable Step) 탐색과 같이 넓은 영역에 탐색 지점들이 펼쳐져 있는 탐색 패턴들을 표현하는데 효과적이다.

각 타입의 패턴레지스터는 동시에 다양한 패턴을 사용 할 수 있도록 4개씩의 특수 레지스터로 저장 하도록 정의하였다. 이를 위해서 패턴 정보 입력을 위한 전용의 입력 명령어를 새로 제안하여 이를 이용하여 패턴 정보를 미리 패턴 레지스터에 저장하면 SAD 연산기가 자동으로 제어되도록 설계하였다.

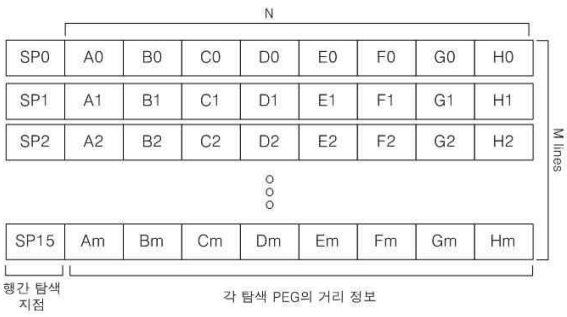

그림 2. 제안하는 패턴레지스터B의 구조
Fig. 2. Structure of proposed pattern register B.

3.2. 제안하는 VBSAD 명령어

제안하는 VBSAD 명령어는 다양한 움직임 추정 연산 환경에 적합하도록 표 2와 같은 다양한 VBSAD 명령어를 지원한다. VBSAD 명령어들은 탐색 시작 지점과 패턴 정보, 연산 크기를 오퍼랜드로 입력 받아 다양한 탐색 패턴과 알고리즘을 처리하도록 정의 하였다. 단순한 연산 구조 덕분에 하드웨어 설계에 많이 사용되는 전역탐색의 경우 탐색 지점이 탐색창에서 순서대로 탐색이 진행되므로 탐색 시작 지점 이외 탐색 패턴에 대한 지정이 필요하지 않다. 하지만 다양한 패턴을 바탕으로 연산하는 고속 탐색 알고리즘을 지원하기 위해서는 탐색창 내에 어느 지점에서나 탐색이 가능해야 한다. 이 때문에 탐색창 정보를 저장하고 있는 외부 메모리에서 원하는 지점에 대한 데이터를 불러와 처리 할 수 있도록 하기 위해서 시작점에 대한 정보를 오퍼랜드로 입력 받도록 정의하였다. 또한 시작 위치에 대한 입력을 직접 값뿐만 아니라, 레지스터로도 입력 받아 단계적 탐색을 하는 고속탐색 알고리즘의 처리가 쉽도록 하였다. 즉, VBSAD 명령 내부에서 이전 연산의 최적 지점을 바탕으로 다음 탐색이 시작되는 지점에 대한 연산을 수행 하여 레지스터에 저장하면 다음 번 연산 시, 레지스터 값을 바탕으로 연산 시작이 가능함으로 시작 위치에 대한 업데이트를 자동으로 지원할 수 있다. 또한, 3.1절에서 설명한 패턴레지스터 타입을 명령어 별로 입력받아 어떤 환경에서도 최적의 패턴으로 연산이 가능하도록 한다.

VBSAD 명령어의 연산 수행은 아래 그림 3와 같이 진행 된다. 외부 참조 영상 메모리에서 시작지점에 대한 정보를 바탕으로 참조할 탐색 영역 데이터를 입력 받는다. 이후 패턴레지스터의 정보에 따라 연산의 유무를 판단한다. 만약 연산 지점이 있다면 입력받은

표 2. 제안하는 ASIP의 VBSAD 명령어
Table 2. VBSAD instructions of the proposed ASIP

명령어	상세기술	참조패턴
SADI	시작위치를 Immediate값을 참조하여 패턴 타입A에 따라 SAD연산	패턴 A
SADR	시작위치를 General register값을 참조하여 패턴 타입A에 따라 SAD연산	패턴 A
SADF	시작위치를 Immediate값을 참조하여 8x32 지점에 대한 Full Search SAD연산	-
SADS	시작위치를 Immediate값을 참조하여 패턴 타입B에 따라 SAD연산	패턴 B

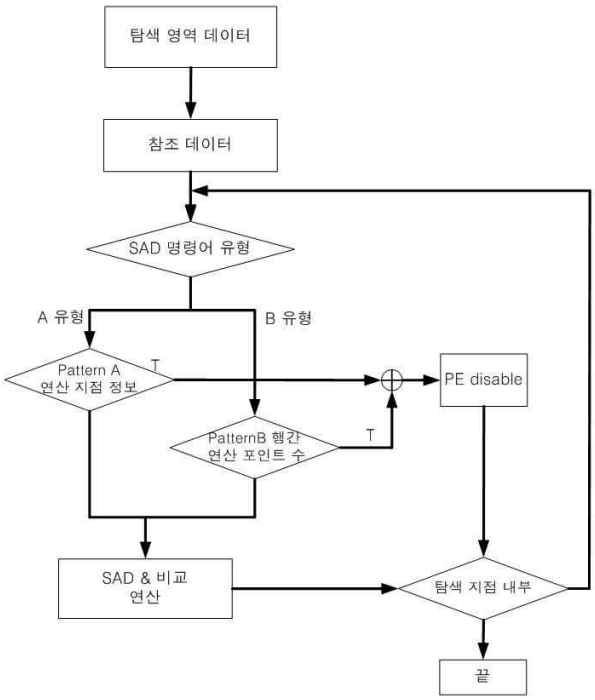

그림 3. 제안하는 VBSAD 명령어 흐름도
Fig. 3. Flowchart of the proposed VBSAD instruction

VBSAD 명령 중 패턴레지스터A와 패턴레지스터B 중 어떤 레지스터를 참조하는 명령어인지를 판단한다. 위 단계의 판단에 따라 패턴레지스터A 명령어 사용 명령의 경우 패턴레지스터A 정보를 확인하여 패턴 비트가 0이면 PE disable 단계로 그렇지 않으면 SAD 연산 단계로 넘어간다. 반면 패턴레지스터B 참조 명령의 경우 탐색할 패턴레지스터B 중 행간 탐색 지점 수 정보에 따라 탐색할 지점 수가 0이 아니면 SAD 연산 단계로 넘어간다.

SAD 연산단계에서는 실제 필요한 연산 지점에 대한 SAD 연산과 비교연산을 수행하여 최적지점을 저장하고 다음 단계로 넘어간다. 마지막으로 현재 탐색 위치가 탐색 지점 내부에 있는지 여부를 판단하고, 다음 탐색 지점이 탐색 영역 내부에 있으면 앞선 과정을 다시 수행하고, 내부에 있지 않으면 연산을 끝내게 된다.

Ⅳ. 제안하는 명령어의 하드웨어 구조

이번 절에서는 Ⅲ절에서 제안한 명령어를 수행하는 전용의 하드웨어 가속기를 제안하고, 구조를 설명한다. 그림 4는 제안하는 하드웨어 가속기의 전체 구조도이다. 기존의 2D SAD 하드웨어 가속기 구조 [10~11]의 단점인 연속된 지점에 대한 처리를 보완할 수 있는 PE

(Processing Element) array구조를 제안하였고 이를 가변 포인트 2D SAD연산기로 명명하였다. 이를 통해 가로 탐색지점을 N개의 PEG (Processing Element Group) 를 사용할 때 최대 N지점에 대해 병렬 처리가 가능하도록 구성되어 있다. 각 지점에 대한 현재 영상 데이터와 참조 영상 데이터가 입력되면 연산에 필요한 각 지점의 데이터를 연산을 담당할 각 PEG에 전달한다. 그리고 각 명령어 별로 참조하는 패턴레지스터의 정보에 따라 연산을 수행할지 안 할지에 대한 결정을 내리고, 이 결과를 바탕으로 각 PEG 를 제어한다. 제어된 PEG에서 연산이 수행되면 각 연산기 별로 4x4 모드에 대한 SAD 연산결과를 동시에 출력한다. 출력된 데이터는 움직임 벡터 생성기와 동기화 되어 그 결과가 비교기로 전달된다. 비교기는 PEG에서 전달된 4x4 모드의 SAD연산 결과를 바탕으로 가변 블록 모드 생성기에서 전체 VBM을 생성한다. 생성된 각 모드들은 비교기 트리에서 각 지점별로 최적지점과 움직임 벡터를 비교하고 그 결과를 저장한다. 각 PEG들은 재구성 가능한 구조를 가지고 있으므로 알고리즘에 따라서 연산양이 적은 경우 적은 수의 PEG만을 사용하여 면적과 전력소모를 낮출 수 있도록 설계하였다.

그림 5는 제안하는 PEG의 가변 포인트 2D SAD 연산기 구조이다. 기존의 2D SAD 구조는 고정된 참조데이터 버퍼로부터 연속된 N개의 지점에 대한 참조 데이터를 입력 받아 처리하여 데이터 재사용율을 높일 수 있다. 이 방법은 데이터 재사용율이 매우 높아 전역 탐색 방법에 매우 효율 적인 방법이다. 하지만, 고속탐색 방법과 같이 탐색지점이 지점간에 서로 떨어져 있는 경우에는 연산기의 이용율이 떨어지는 단점을 갖고 있다. 따라서, 제안하는 가변 포인트 2D SAD 구조는 데이터 재사용율을 높이면서도 불연속 탐색 지점에 대한 처리가 가능하도록 제안하였다.

그림 5를 참조하면, 외부 탐색창 메모리로부터 현재 매크로 블록 처리에 필요한 참조데이터들이 참조 데이터 버퍼로 입력된다. 이후, 각 데이터들은 시프트 어레이를 통해 각 지점별로 필요한 만큼 이동 시켜둔다. 다음 다중화 유닛은 실질적으로 탐색할 데이터 개수를 바탕으로 각 PEG에 시프트 어레이를 통해 이동된 참조데이터들을 선별적으로 입력하여 SAD 연산을 수행하게 된다. 이때 각 유닛에 필요한 제어 신호들은 오퍼랜드로 입력 받은 패턴 정보와 사이즈에 의해서 결정된다. 앞서 그림 2에서 설명한 바와 같이 패턴레지스터 B를 사용하는 명령어의 경우 shift enable을 켜고, 연산을 수행할 각 지점의 거리정보를 Ref select signals로 하여 각 PEG 별로 선택된 데이터를 입력하고, 탐색지점 수를 PEG enable로 하여 연산을 수행하게 된다. 또, 패턴레지스터A를 이용하는 명령어의 경우 Shift enable은 끄고, Ref select signal은 첫 지점부터 연속된 N지점으로 설정하고, 마스크 방식의 패턴레지스터 A 정보를 PEG enable 로 하여 연산을 수행함으로써 기존의 2D SAD 연산 방식의 장점을 갖고서 불연속 지점에 대한 연산 또한 처리 할 수 있도록 하였다.

그림 6에서 볼 수 있듯이 하나의 PEG는 128개의 PE

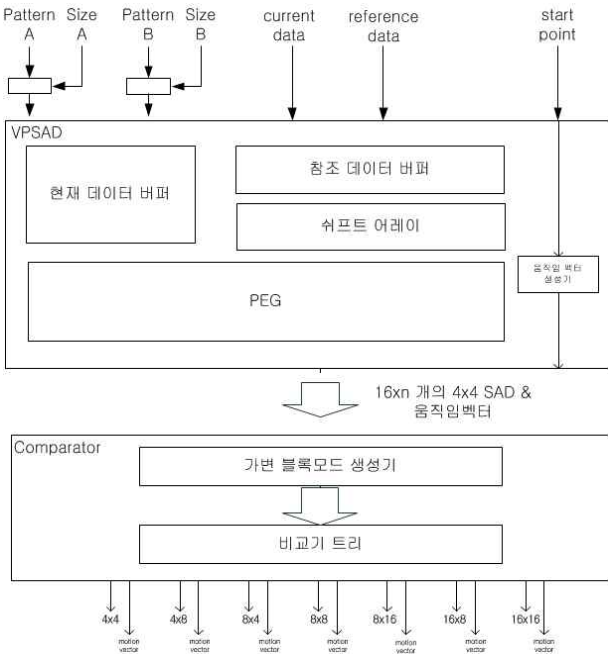

그림 4. 제안하는 IME 구조
Fig. 4. Proposed IME architecture.

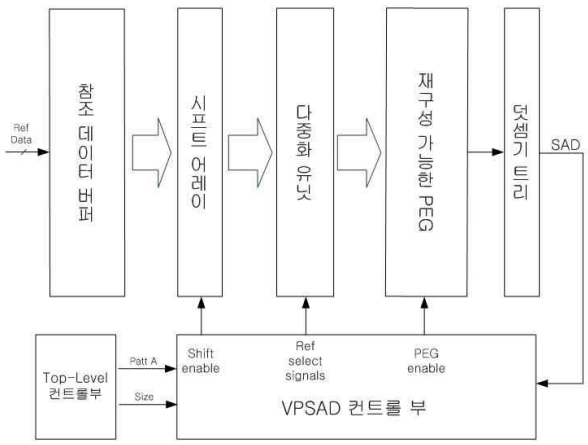

그림 5. 제안하는 가변 포인트 2D SAD 구조도
Fig. 5. Proposed Variable Point 2D SAD architecture.

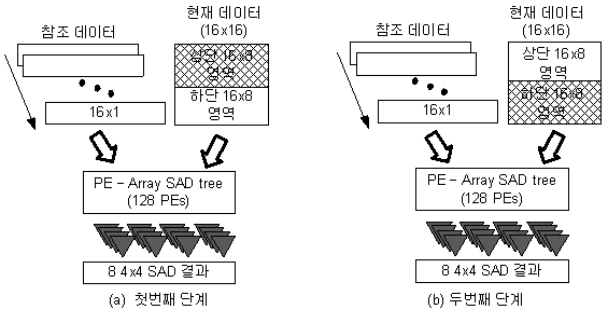

그림 6. 제안하는 PEG 아키텍처
Fig. 6. Proposed PEG architecture.

로 구성하였다. 따라서 동시에 4x4 모드에 대한 8개의 SAD 갑슬 생성해 낸다. 생성된 결과는 SAD PEG 뒤에 위치한 비교기의 입력으로 들어간다. 제안하는 가변 포인트 2D SAD 구조는 제안하는 VBSAD 명령어의 기본 처리 단위인 16x16 단위를 처리하기 위해서 2 사이클에 나누어 첫 사이클에 16x16의 상위 16x8 픽셀을, 두 번째 사이클에 하위 16x8 픽셀을 처리하였다. 이는 비록 제어기의 복잡도와 연산 사이클이 증가하더라도 PE의 개수를 줄여 제안하는 가변 포인트 2D SAD 구조의 하드웨어 복잡도를 낮추기 위함이다.

V. 구현 및 성능 비교

제안한 정화소 움직임 추정 명령어들은 LISA 언어 기반의 Synopsys Processor Designer툴을 이용하여 시뮬레이션을 하고 성능을 검증하였다. 하드웨어 구조는 Verilog HDL 언어를 이용하여 하드웨어로 구현하였고 Mentor의 ModelSim을 이용하여 동작을 검증하였다. 그리고 Synopsys의 Design Compiler를 이용하여 IBM 90nm 표준 셀 라이브러리로 합성한 결과, SAD 연산기의 PEG하나 당 25,500 개의 게이트를 가지며, Comparator는 약 109,720 개의 게이트를 가진다. 최대 동작 주파수는 약 232MHz로 측정되었다.

표 3은 서론에서 소개한 ASIP들 간의 대표적인 고속 탐색알고리즘인 4 스텝 탐색과 다이아몬드 탐색에 대한 코드 사이즈 및 연산 사이클 비교를 보여준다. [8]과 [12]는 단일 연산 명령어를 다수 사용하여 소프트웨어적인 처리가 많아 코드 사이즈가 제안하는 ASIP보다 크며 반복적인 연산 처리로 인해 매크로블록 당 연산 시간도 더 많이 필요한 것을 알 수 있다. [9]는 다이아 소요 되는 반면 제안하는 ASIP은 192 사이클이 소요된다. 이때 하드웨어 크기를 비교해보면 Virtex-4 SX35로

표 3. 제안하는 ASIP과 다른 ASIP의 코드 사이즈 및 연산 사이클 비교
Table 3. Code size and computation cycle comparisons between the proposed ASIP and other ASIP.

	4스텝탐색		다이아몬드탐색		H/W size
	코드 크기	사이클 수	코드 크기	사이클 수	LUT
[8]	365	5248	460	4352	-
[12]	88	722	84	660	-
[9] (PEG 2개)	-	-	-	287	3805
Proposed (PEG 2개)	37	100	32	192	6594

합성한 결과 [9]는 3,805 LUTs를 제안한 ASIP은 이보다 두 배 정도인 6,594 LUTs 가 나왔다. 이는 제안하는 ASIP은 데이터 재사용을 위하여 총 31지점의 처리를 위한 참조 데이터 버퍼 396 바이트, 현재 데이터 버퍼 256 바이트를 두고 있어 필요 메모리가 포함된 수치인 반면, [9]는 필요 메모리를 제외한 연산기 사이즈만 포함된 수치이다. 또한, 제안하는 ASIP이 모든 VBM을 지원하기 위하여 4x4 단위의 연산 결과를 덧셈 트리를 이용해 더하고 모든 VBM의 SAD 값을 저장하는 반면, [9]는 8x8, 16x16 두 모드만 연산함으로 제안하는 ASIP의 하드웨어 사이즈가 [9] ASIP에 비하여 커질 수밖에 없다. 이를 통해 볼 때 제안하는 ASIP은 병렬 연산이 가능한 SAD 명령어를 통해 더 적은 코드 사이즈와 더 적은 연산 사이클로 움직임 추정 연산을 수행함을 알 수 있다.

표 4는 제안 하는 명령어와 하드웨어 구조를 사용하는 ASIP과 기존 [11] 논문이 제안한 ASIP과의 성능을 DVSS A1 알고리즘, UMHS 알고리즘, 그리고 다이아몬드 탐색 알고리즘에 대하여 연산 시간 및 하드웨어 크기를 비교하였다. 다이아몬드 탐색의 경우 두 명령어 모두 8번의 명령어로 처리가 가능하고 이때 제안하는 구조가 기존 [11]보다 연산 사이클 면에서 14%정도 느린 것을 알 수 있다. 이 것은 제안하는 구조가 기존 구조보다 PEG당 PE수를 절반을 사용하면서 나타나는 효과이다. 하지만, 앞서 설명한 바와 같이 DVSS 알고리즘이나 UHMS와 같이 넓은 패턴 영역에 탐색 지점이 흩뿌려진 알고리즘에 대해서 가변 포인트 2D SAD 구조를 사용하여 한 명령어 당 처리 지점 수를 늘림으로

표 4. 제안하는 ASIP과 [11]의 연산시간 및 하드웨어 크기 비교
Table 4. Computatioon cycle and hardware size comparsions between the proposed ASIP and [11]

	DVSS A1		UMHS		DS		Hardware	
	명령어수	사이클수	명령어수	사이클수	명령어수	사이클수	게이트수	MHz
[11]	55	1589	11	279	8	168	614K	237.5
Proposed	30	1248	7	232	8	192	453K	232.0

표 5. 제안하는 ASIP과 ASIC과의 하드웨어 비교
Table 5. Hardware comparsions between the proposed ASIP and other ASICs.

IME	[10]	[13]	Proposed	
tech(㎛)	UMC 0.18	TSMC 0.18	IBM 0.09	
Gate Count	305k	481k	453k	271k
# of PEG	4	8	8	4
Resolution	720P	720p	1080p	
Real time freq(MHz)	81/108MHz	110.5	188	

써 전체 연산 사이클을 줄이는 것을 볼 수 있다. DVSS 알고리즘의 경우 약 21%, UMHS 알고리즘의 경우 약 17% 감소시킬 수 있었다. 즉, 제안하는 ASIP은 [11] 논문의 ASIP에 비교하여 하드웨어 사이즈 측면에서 26%, 연산 속도 측면에서 18%를 줄였다.

표 5에서 보이는 바와 같이 제안하는 ASIP은 [10][13]의 ASIC보다 하드웨어 사이즈가 작다. 보통은 ASIP의 크기가 더 크지만 언급한 바와 같이 제안하는 ASIP의 PEG가 기존의 PEG보다 PE의 개수가 절반이기 때문에 ASIC보다 크기가 작다. 제안하는 AISP의 동작 주파수는 188MHz이다. 이는 다른 ASIC의 동작속도인 108MHz과 110.5MHz 보다는 큰 값이나 1080p 영상의 실시간 처리를 충분히 지원 할 수 있다.

VI. 결 론

본 논문은 다양한 비디오 압축 알고리즘을 위한 ASIP전용의 정화소 움직임 추정 명령어와 하드웨어 구조를 제안한다. 제안한 정화소 움직임 추정 명령어는 다양한 알고리즘을 효과적으로 지원하는 VBSAD 명령어들로 구성되어 있다. 제안한 ASIP은 이 전에 제안된 ASIP과 비교하여 보다 적은 연산기를 가지고 효율적으로 연산이 가능하며 특히 최근 제안되고 있는 고속 탐색 알고리즘에 적합한 SAD 명령어와 가변 포인트 2D SAD 구조를 통해 보다 더 효율적인 연산을 제공한다. 또한, 본 논문에서 제안한 ASIP의 구조는 다양한 비디오 알고리즘을 사용하는 모든 분야에 사용될 수 있다. 본 구조는 현재 상용 Very Long Instruction-set Word (VLIW) 코어 등에 비해 작은 하드웨어를 가지므로 저전력으로 시스템을 구성하기에 용이하며, 비디오 시스템을 위한 임베디드 SoC 환경에 적합하다. 또한 영상 압축 환경뿐만 아니라 최근 디지털TV 시장에서 중요한 기술적 이슈를 가지고 있는 프레임 속도 변환 (Frame Rate Conversion; FRC)과 같이 움직임 추정을 요구하는 시스템에 쉽게 접목할 수 있다.

참 고 문 헌

[1] Joint Video Team (JVT) of ISO/IEC & ITU-T VCEG, “Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification,” ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC, July 2004.

[2] T. Wiegand, G.J. Sullivan, G. Bjontegaard and A. Luthra, “Overview of the H.264/AVC video coding standard,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 560-576, July 2003.

[3] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, “Video coding with H.264/AVC: Tools, Performance, and Complexity,” *IEEE Circuits and Systems Magazine 1*, pp. 7-28, Apr. 2004.

[4] Jung H. Lee, Sung D. Kim, and Myung H. Sunwoo, “ASIP Instructions and Their Hardware Architecture for H.264/AVC,” *Journal of Semiconductor Technology and Science (JSTS)*, vol.5, no.4, pp. 237-242, Dec 2005.

[5] TMS320C6000 CPU and Instruction Set Reference Guide, Texas Instruments Inc., Dallas, TX, 2000

[6] Intel Architecture Software Developer’s Manual Volume 2: Instruction Set Reference, Intel Inc., 1999.

[7] Intel® SSE4 Programming Reference, Intel Inc., July 2007

[8] Momcilovic, S.; Roma, N.; Sousa, L., “An ASIP approach for adaptive AVC Motion Estimation,” *Research in Microelectronics and Electronics Conference*, pp165-168, 2007.

[9] J. L. Nunez-Yanez, T. Spiteri, and G. Vafiadis, "Multi-standard reconfigurable motion estimation processor for hybrid video codecs," in *IET Computers & Digital Techniques*, Vol. 5, Iss. 2, pp. 73 - 85, 2011.
[10] T. C. Chen, and et al., "Analysis and Architecture Design of an HDTV720p 30 Frames/s H.264/AVC Encoder," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no.6, pp. 673-688, June 2006.
[11] Hee Kwan Eun, Sung Jo Hwang, Myung Hoon Sunwoo, Yung Hwan Kim, Hi Seok Kim, "Integer-pel Motion Estimation Specific Instructions and their Hardware Architecture for ASIP", in. Proc. *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2011.
[12] Konstantinos Babionitakis, Gregory A, Doumenis, George Georgakarakos, George Lentaris, Kostantinos Nakos, Dionysios Reisis, Ioannis Sifnaios, Nikolaos Vlassopoulos, "A real-time motion estimation FPGA architecture", *Journal of Real-Time Image Processing.*, vol. 3, pp. 3-20, 2008.
[13] Yiqing Huang, Qin Liu, and Takeshi Ikenaga, "Spatial feature based reconfigurable H.264/AVC integer motion estimation architecture for HDTV video encoder," in *Proc. International Conference on Digital Signal Processing (DSP)*, July 2009, pp. 1-6.

저 자 소 개

방 호 일(학생회원)
2009년 아주대학교 정보통신대학 전자공학 학사 졸업.
2011년 아주대학교 정보통신대학 전자공학 석사 졸업.
2011년 현재 LG전자(주) 연구원

<주관심분야 : 멀티미디어 코덱, 멀티미디어 신호처리, 멀티미디어용 SOC설계, 저전력 반도체 설계>

선우 명 훈(정회원)-교신저자
1980년 서강대학교 전자공학 학사 졸업.
1982년 한국과학기술원 전자공학 석사 졸업.
1982년～1985년 한국전자통신연구소(ETRI) 연구원
1985년～1990년 Univ. of Texas at Austin 전자공학 박사.
1990년～1992년 Motorola, DSP Chip Division (미국)
1992년～1996년 아주대학교 전기전자공학부 조교수.
1996년～2001년 아주대학교 전자공학부 부교수
2001년～현재 아주대학교 전자공학부 교수
2010년～현재 대한전자공학회 반도체 소사이어티 수석 부회장
2011년～현재 IEEE CASS Board of Governor
<주관심분야 : SoC 설계, VLSI Architecture, 통신 및 멀티미디어 ASIP 설계, 저전력 설계>