濟州大學校 尖端技術硏究所 論文集, 第15卷, 1號, pp.?-?, 2004. 12
Cheju Nat'l Univ. Res. Inst. Adv. Tech. Jour., Vol.15, No.2, pp.?-?, 2004. 12

개선된 MRME 알고리즘을 이용한 H.264 움직임 추정기 설계 및 FPGA 검증

진군선*· 강진아*· 임재윤**

H.264 motion estimation unit design and FPGA verification using enhanced MRME algorithm

Goon-Seon Jin*, Jin-Ah Kang*, Jea-Yun Lim**

ABSTRACT

This paper presents an architectural enhancement to reduce the data load of the Multi-Resolution motion estimation. Our approch is based on eliminating unnecessary data load using memory reuse. New hardware architecture for integer-pel ME(motion estimation) dedicated to H.264/AVC is proposed. The proposed architecture supports all 7 modes (16x16, 16x8, 8x16, 8x8, 8x4, 4x8, and 4x4) for variable block size ME. The features of our design are 2-D PE(processing element) array and SAD merging scheme. A pipelined and shared datapath architecture for motion estimation unit are designed to improve the system performance at the reduced hardware complexity.

Key Words : H.264/AVC, Verilog HDL, motion estimation, Xilinx FPGA

Ⅰ. 서 론

H.264/AVC는 ITU-T와 ISO가 함께 표준화 과정을 진행시켜 얻어낸 새로운 비디오 압축 표준으로 영상시스템과 관련된 모든 영역에의 적용이 가능한 압축기술로써 차세대 영상시스템을 주도할 수 있는 획기적인 코덱(codec)이라 할 수 있다. 파일 압축률이 높아 1Mbps 이하의 인터넷 망을 통해서 DVD 수준의 동영상을 전달하면서도 네트워크 자원을 덜 차지하는 장점이 있다[1].

H.264는 현재 우리나라 위성 DMB와 지상파 DMB의 동영상 코덱으로 채택되었으며 앞으로 HD DVD 플레이어, 디지털 TV, 디지털 캠코더, 휴대 이동 단말기에도 동영상 코덱 표준으로 채택될 가능성이 매우 높다. 현재 H.264 영상 코덱을 DSP(digital signal processor)등으로 구현하려는 시도는 많이 있지만 대부분 200~300MHz의 고주파에서 동작하는데 반해 ASIC으로 구현하였을 시에는 30MHz 이하에서도 실시간 영상 압축이 가능하여 전력 소모에 민감한 차량용 또는 이동 단말기에서 시스템 경쟁력을 매우 높일 수 있다. 또한 ASIC으로 구현시에는 성능이 DSP로 구현한 시스템보다 매우 안정적으로 동작하여 이동성이 많고 열악한 환경에서 동작해야하는 모바일 멀티미디어 단말에 특히 유리하다고 할 수 있다. 그리고 컨버전스화, 경박단소화를 만족해야하는 현대의 모바일 멀티미디어 단말에는 범용성을 위

**제주대학교 통신컴퓨터 공학부, 첨단기술연구소
Faculty of Telecommunication and ComputerEng., Research of Advanced Technology, Cheju Nat,l Univ.
*제주대학교 대학원
Graduate School, Cheju Nat,l Univ.

해 꼭 필요하지 않은 기능이 많이 내장된 범용 DSP보다는 이동 단말에 최적화된 ASIC이 절대적으로 유리하다.

본 논문에서는 H.264 영상압축 AISC에서 성능을 좌우하는 움직임 추정기를 구현하는데 있어 효율적인 구조를 갖고 성능 향상된 모듈을 설계하는데 목표를 둔다. 효율적인 설계를 위하여 움직임 추정을 위한 알고리즘을 분석하고 가장 적절한 알고리즘을 선택한다.선택된 알고리즘의 핵심적인 기능을 분석하고, 이를 구현하기 위해 적합한 움직임 추정기 전체 구조를 제안한다. 그리고 전체 구조에 속하는 여러 중요 모듈 설계 구조에 대한 연구와 움직임 추정 알고리즘을 가장 적합하게 구현할 수 있는 구조에 대해 심도있게 분석한다. 또한 움직임 추정기의 내부 구조와 이를 구현하기 위한 설계 방법 및 테스트 환경을 살펴본다. 보드레벨에서의 검증은 EISC(extensible instruction set computing) 방식의 SE3208 코어 MCU 연결하여 FPGA(field programmable gate array)로 검증하고 그 결과를 비교·검토한다.

본 논문의 구성을 살펴보면 Ⅱ장에서는 움직임 추정을 위한 다양한 알고리즘에 대해서 설명을 기술한다. Ⅲ장에서는 H.264 움직임 추정기를 Verilog HDL로 기술하여 시뮬레이션 결과 및 합성 결과를 제시한다. Ⅳ장에서는 FPGA 상에서 구현하여 검증된 결과를 제시한다. 또한 설계된 움직임 추정기를 기존의 움직임 추정기와 비교하여 어느정도의 성능 향상이 있는지 알아본다, 마지막으로 Ⅴ장에서는 본 논문의 결론을 맺는다.

Ⅱ. 움직임 추정기 알고리즘

동영상 시스템에서의 압축 부호화를 위해서는 동영상에 존재하는 4가지 중복성, 즉 신호 성분간에 존재하는 중복성, 화면 내에 존재하는 공간적 중복성, 화면간에 존재하는 중복성, 그리고데이터의 통계적 발생 확률에 존재하는 통계적 중복성을 효과적으로 제거해야 한다. 화면 내에 존재하는 공간적 중복성은 변환부호화와 양자화 과정을 통해서 제거하고 화면간에 존재하는 시간적 중복성은 움직임 추정/보상 기법에 의해 제거된다. Fig. 1은 전형적인 동영상 부호화기의 블록도를 나타낸 것이다. 주요 블록은 주파수 변환단(transform unit), 움직임 추정단(motion estimation unit), 움직임 보상단(motion compensation unit), 허프만 부호화단(entropy coding unit)이다. R은 참조 블록이며 S는 탐색 영역 블록을 말한다.

동영상 부호화기의 압축 과정에서 시간적 중복성을 제거하기 위해 이전 프레임의 데이터를 이용하여 움직임 추정 및 보상을 수행하고, 이때 추정된 움직임 벡터(motion vector : MV)에 의해 보상된 영상과 원 영상과의 차 신호를 부호화하여 전송하게 된다. 이 방법은 동영상 부호화에서 가장 높은 압축률을 가져오지만 많은 계산량으로 인하여 부호화기의 전체 성능에 결정적인 영향을 미치게 된다.

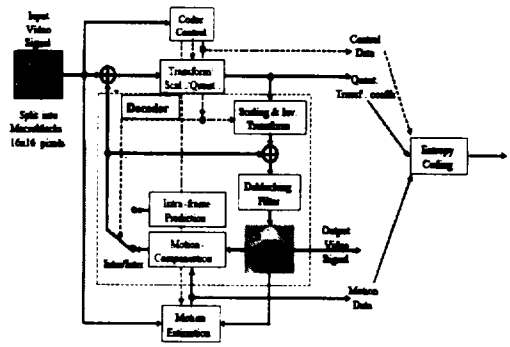

Fig. 1. Block diagram of moving picture encoder

Fig.2는 움직임 추정 과정을 간략하게 나타낸 그림이다. 움직임 추정은 이전 화면에서 현재 화면을 예측하는 방법이므로 이 과정은 기준블럭과 이전 화면 블러간의 화소값 차이를 계산하여 그 값이 가장 작은 블록을 택하는 방식으로 진행된

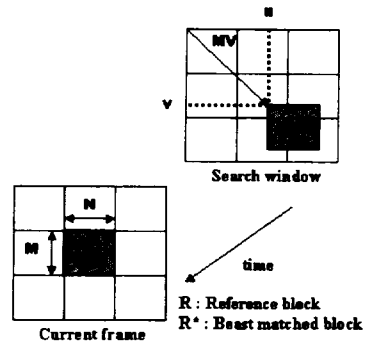

Fig. 2. Motion estimation

다. 이때 행해지는 연산은 보통 회로 구현에 있용이한 SAD(sum absolute difference)가 사용한다. R은 참조 블록이며 S는 탐색 영역 블록을 말한다

Ⅲ. H.264 움직임 추정기 설계 구조

Fig. 3은 구현할 움직임 추정 알고리즘을 나타낸다. 레벨2에서는 움직임 추정기로의 입력 영상을 수평, 수직 방향으로 4:1로 샘플링한 영상을 이용하여 [-2, +1] 탐색영역에서 4x4 블록 단위로 움직임 추정을 수행한다. 레벨2에서 탐색을 수행하여 SAD(sum absolute difference)가 최소가 되는 2점을 구한다. 그리고 인접한 매크로블럭의 움직임 벡터들의 중간값을 이용하여 한 점을 구한다. 이렇게 구해진 3개의 점을 이용한다. 레벨1에서는 원래의 영상을 수평, 수직으로 2:1 샘플링한 영상에서 위의 3개의 탐색 중심점을 중심으로 8x8 블록 단위로 [-2 +1]의 부분을 탐색한다. 탐색 수행 후 정합 기준이 되는 최소가 되는 하나의 점을 선택하여 하위 레벨의 탐색 중심점으로 이용한다[2].

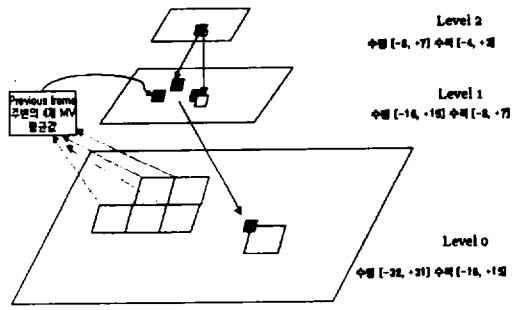

Fig. 3. Enhanced MRME algorithm

본 논문에서 제안하는 움직임 추정기 전체 구조는 Fig. 4와 같다. 주요 블록으로는 AMBA Bus와 인터페이스 되는 AMBA I/F, AMBA Bus를 통해 전송되는영상 데이터를 3개의 레벨로 분리 시키는 해상도 변환기(resolution converter), 영상 데이터를 저장하는 RAM 배열기(ram_array), RAM을 제어하는 메모리 제어기(ram controller), PE_ARRAY 블록의 입력값을 8x8 형태로 변환하는 레지스터 배열기(reg_array), 영상데이터의 차의 절대값을 구하는 PE_ARRAY, 각 SAD 합을 비교하여 최소가 되는 움직임 벡터를 추출하는 비교단(compare unit), 움직임 추정기 전체 제어를 담당하는 전체 제어기(MRME controller) 이다.

레벨0에서는 원래의 영상에서 16x16 매크로 블록 단위로 수평, 수직 [-2, +1]의 부분을 탐색한다. H.264/AVC의 다양한 블록크기를 지원하기 위해 4x4 블록 단위로 정합기준을 구하고 이를 더하여 4x4 블록보다 더 큰 블록에 대한 정합기준을 계산하는 방식을 이용한다.

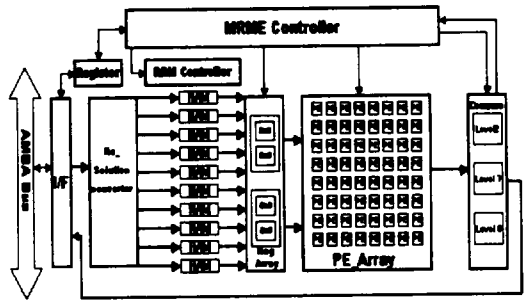

Fig. 4 Motion estimation hardware architecture

메모리 제어기는 탐색 영역 데이터와 참조 영상 데이터를 저장하는 RAM을 위한 제어기이다. 제어 신호와 어드레스 값, 데이터 값등을 발생시켜 RAM에 데이터를 저장하기도 하고 저장된 데이터를 다시 읽어 오기도 한다. 영상 시스템은 특성상 메모리에 데이터를 저장하고 읽어 오는 기능을 수행할 때가 빈번하며 데이터양 또한 대용량일 경우가 많다. 특히 영상 시스템에서 움직임 추정기는 여러 프레임간의 움직임 추정을 수행하므로 저장하고 읽어오는 데이터의 이동이 가장 빈번하다. 이러한 움직임 추정기에서 메모리 컨트롤러를 최적화 하지 않는다면 움직임 벡터를 계산하는 것 보다 계산할 영상 데이터를 읽어 오는데 필요한 클럭이 과도하게 지연되여 움직임 추정기의 성능을 상당 부분 저하시키게 된다. 따라서 본 논문에서는 움직임 추정기 성능 향상을 위한 메모리 재사용(memory reuse) 방법을 제안하고 이를 구현 하고자 한다. Fig. 5는 메모리 재사용 방법을 이용하여 [-2, +1] 탐색에 해당하는 16개 탐색 순서와 탐색을 위한 데이터 흐름을 나타낸 것이다. 메모리 재사용을 위해 4가지의 입

력 상태가 있는데 다음과 같다. 먼저 8x8 블록의 입력을 모두다 받아 들이는 주소 증가(inc_add) 상태인데 Fig. 5에서 처음 8x8 블록의 데이터에 해당하는 사각형 그림이다. 다음은 오른쪽으로 이동하면서 행(row)방향으로 8개의 입력값을 받아 들이는 오른쪽_주소(right_add) 상태인데, 좌측으로 한 픽셀씩 이동하면서 (-1, -2), (0, -2), (1, -2) 위치에서의 움직임 벡터를 찾게된다. 다음은 (1, -1) 탐색점의 움직임 벡터를 찾게 되는데 Fig. 13에서 열(column)방향으로 8개의 입력값을 받아 들이는 아래쪽_주소(dw_add) 상태이다. 또한 왼쪽으로 이동하면서 행(row)방향으로 8개의 입력값을 받아 들이는 왼쪽_주소(left_add) 상태가 있는데 우측으로 한 픽셀씩 이동하면서 (0, -1), (-1, -1), (-2, -1) 위치에서의 움직임 벡터를 찾게 된다.

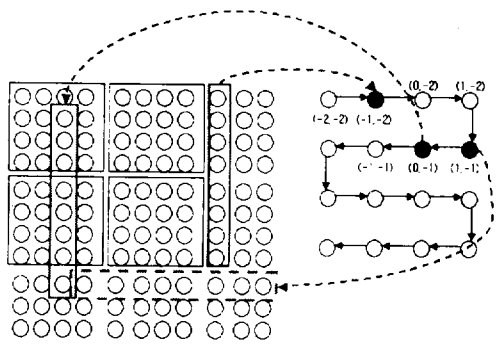

Fig. 5. Concept of memory reuse

Fig. 6는 PE_ARRAY unit를 나타낸 것이다. 그림에서 보는 것과 같이 전에 설명한 PE 64가 배열(array) 형태로 내장되어 있다. 입력값으로는 레지스터 배열기(reg_array)에서 출력된 8x8 매크로 블럭 단위의 탐색 영역 데이터와 참조 영상 데이터 128개 이다. 단일 클럭에서 계산된 64개의 차의 절대값의 합은 4부분으로 나뉜 각 블록의 합으로 출력된다. 즉, 출력은 PE 16가 모인 pe_sad_sum0, pe_sad_sum1, 2, 3 이다.

레지스터 배열기 모듈은 RAM으로부터 이전영상 및 참조 영상을 받아 8x8 형식의 데이터로 만들어 PE_ARRAY로 전달하는 역할을 한다. 레지스터 배열기 모듈은 3개의 sub-block으로 나뉘는데 Fig.7.와 같이 크게 RAM_I/F 블록, PREV_REG CUR_REG 블록이 있다. RAM_I/F 모듈은

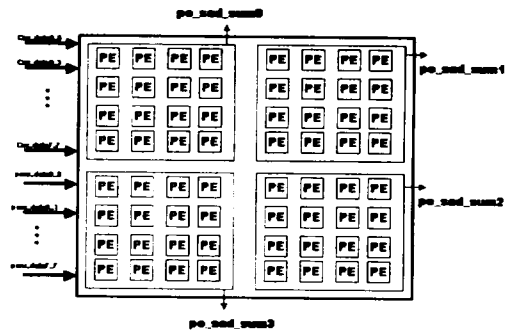

Fig. 6. PE_Array architecture

RAM_ARRAY로부터 이전 영상 및 참조 영상을 받아 이전영상 레지스터 및 현재영상 레지스터에 데이터를 전달하는 역할을 한다. 이전영상 레지스터 모듈은 RAM_IF로부터 이전 영상 데이터를 받아 8x8 형식의 데이터로 만들어 PE_ARRAY로 전달하며 마찬가지로 현재영상 레지스터도 RAM_I/F로부터 참조 영상을 전달받아 PE_ARRAY로 전달한다.

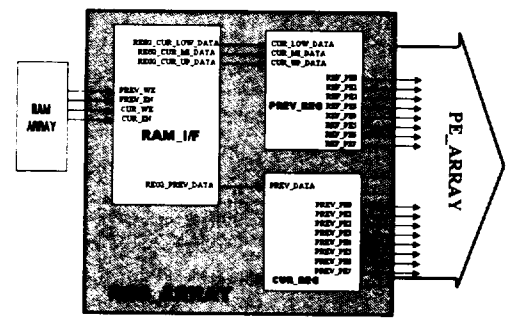

Fig. 7. Block diagram of REG_ARRAY

Fig. 8는 H.264 움직임 추정기를 합성한 결과이다. 6개의 주요 블록이 있는데 제일 앞단의 모듈이 움직임 추정기 전체 제어를 담당하는 전체 제어기(mrme controller)이며 그 다음은 해상도 변환기(resol_cnv)이다. 그리고 메모리 제어기(ram_controller), 10개의 RAM으로 구성된 램 배열(ram_array), 다음이 레지스터 배열기(reg_array), 그리고 PE_ARRAY, 마지막으로 비교단(compare Unit) 모듈이다. 합성은 XILINX의 ISE 프로그램에 포함된 XST(xilinx synthesis tool)를 이용하여

합성하였다. 타켓 디바이스는 XILINX XCV600 -HQ240C(660KGates, 166개의 I/O)로 하였다. 10개의 RAM은 XILINX ISE 프로그램에 포함된 CORE GENERATION 툴을 사용하여 구현하였다.

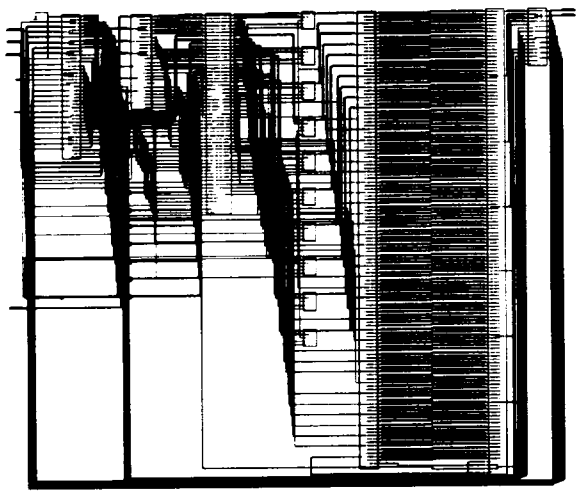

Fig. 8. Synthesis result of H.264 motion estimation

Ⅳ. FPGA 검증 및 성능 비교

설계된 H.264 움직임 추정기를 보드 레벨에서 검증하기 위해서 다음과 같이 검증 환경을 구성하였다. 움직임 추정기 IP(intellectual property)는 Xilinx FPGA에 하드웨어 프로그래밍하였고 움직임 추정기를 제어하고 영상 데이터를 전달하며 결과값을 읽어오는 기능은 Jupiter MCU가 담당하도록 하였다. Fig. 9는 Xilinx FPAG 보드와 Jupiter MCU 보드 사진이다. 검증 방법은 이전 영상 데이터와 현재 영상 데이터를 포함하고 MCU를 제어하기 위한 펌웨어를 시리얼 포트1(RS-232C)를 통하여 MCU에 전달하면 MCU는 탐색 영역의 이전 프레임 데이터와 찾고자 하는 현재 프레임의 매크로블럭 영상 데이터를 FPGA로 전달한다.

개선된 MRME 알고리즘을 FPGA에 구현하고 원하는 기능을 수행하는지 검증하였다. H.264 움직임 추정기는 세 개의 해상도 레벨에서 각각 움직임 벡터를 찾는다. Level2에서는 두 개의 움직임 벡터를 찾게되며 레벨1에서는 레벨2에서의 움직임 벡터 포인터를 중심으로 최소가 되는 움직임 벡터 하나를 구하게 된다. 마지막으로 레벨0에서는 레벨1의 움직임 벡터 포인터를 중심으로 최소가 되는 움직임 최종적으로 찾게 된다.

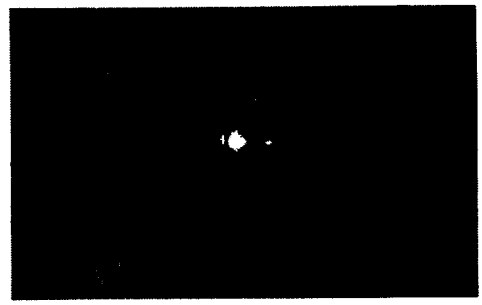

Fig. 9. Interface architecture

Fig. 11은 하이퍼터미널을 이용하여 레벨2 움직임 벡터값을 FPGA에서 읽어온 데이터를 나타낸다. 레벨2에서의 가장 작은 SAD를 갖는 움직임 벡터로 (5,7)가 생성되었고 Fig.10의 시뮬레이션 결과와 FPGA에서 얻어진 결과와 일치함을 확인할 수 있었다.

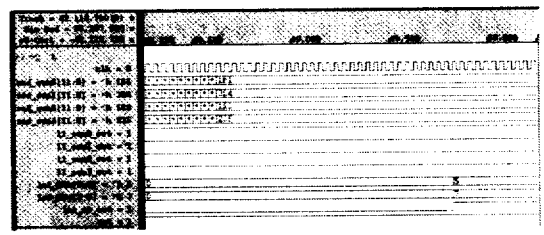

Fig. 10. Simulation result of Level1 compare

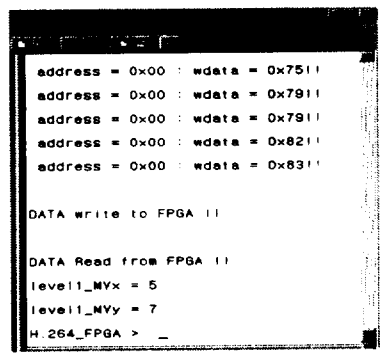

Fig. 11. Level1 simulation result value

제안된 알고리즘과 구조로 설계된 H.264 움직임 추정기는 CIF급(352x288), 30frames/sec 이미지를 적용하였을 때 데이터를 읽어오는 데이터 로딩(loading)에는 1656 clock이 소요되었고 데이터 처리(processing)에는 444clock이 필요하였다. Fig. 12에서는 하나의 MB을 처리하는데 필요한 클럭 수를 비교하였다. 효율적으로 설계된 움직임 추정기일수록 클럭수는 적다. 그림에서 알 수 있듯이 FS1이 클럭 수 1792로 가장 성능이 좋으며 제안된 움직임 추정기는 두 번째 성능인 2100을 필요로 하였다. Fig. 13는 CIF급 이미지 처리의 기준이 되는 36.5MHz에서 처리할 수 있는 프레임 수를 비교한 것이다. 기준이 되는 36.5MHz는 초당 30프레임의 CIF급 이미지를 처리하는 필요한 대역폭으로서 30프레임을 처리하는 필요한 클럭수가 36.5MHz 가 넘으면 사실상 실시간 복원이 어렵게 된다. 그림에서 알 수 있듯이 제안된 움직임 추정기는 36.5MHz에서 43개의 프레임을 처리할 수 있어 초당 30 프레임을 넉넉히 처리할 수 있음을 알 수 있다. 면적과 클럭 수는 trade off 관계에 있는데 면적을 작게 만들면 클럭 수는 증가되어 성능이 저하된 움직임 추정기로 구현되며 클럭 수를 줄여 성능을 향상시키면 면적은 증가되는 현상이 발생하게 된다.

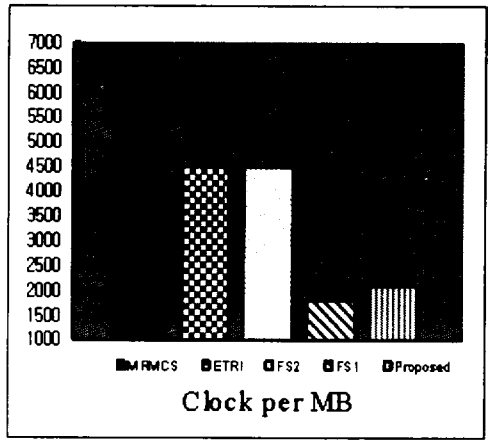

Fig. 12. Histogram of clock per MB

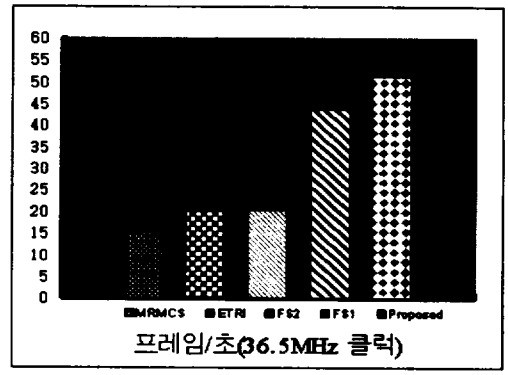

Fig. 13. Histogram of frames per second

V. 결 론

제안한 움직임 추정 구조를 Verilog HDL로 기술하고 FPGA상에서 구현하여 그 기능을 검증하였다. H.264를 위한 다해상도 움직임 추정 알고리즘의 구조를 기술하고 여러 기능을 추가하여 하드웨어로 제안된 모듈을 구현하였다. 이를 위해 다른 몇 가지의 움직임 추정 알고리즘과 제안된 구조의 성능을 비교하여 제안한 움직임 추정기가 동작 주파수, 처리 속도 면에서 우수한 성능을 나타내었다. 제안된 움직임 추정기는 동작 주파수가 25MHz일때 CIF급(352x288), 초당 30 프레임 영상에 대해서 정화소 단위 움직임 추정을 처리할 수 있다. 또한 다해상도 움직임 추정기는 실시간으로 동영상 부호화가 가능한 성능을 가지고 있기 때문에 H.264 움직임 추정기를 하드웨어로 설계하기에 우수한 구조라고 할 것이다.

참고문헌

1) A. Tamhankar and K. R. Rao, 2003, An Overview of H.264/MPEG-4 PART 10, 4th EURASIP conference.

2) Jae Hun Lee, Kyoung Won Lim, Byung Cheol Song, and Jong Beom Ra, 2001, A Fast Multi-Resolution Block Matching Algorithm and its LSI Architecture for Low Bit-Rate Video Coding, IEEE Trans. on Circuit and Systems for Video technology, vol. 11.