JPEG 베이스라인 디코더용 헤더 파서 모듈 구현

노 시 찬 · 손 승 일 · 오 승 호* · 이 민 수

한신대학교, *칩스브레인 글로벌

Implementation of Header Parser Module for JPEG Baseline Decoder

Si-chan Noh · Seung-il Sonh · Seung-ho Oh* · Min-soo Lee

Hanshin University, *CHIPSBRAIN GLOBAL

E-mail : jsichan@nate.com

요 약

JPEG(Joint Photographic Expert Group)은 손실 압축 기법을 사용하여 데이터 양을 20:1 이상으로 현저히 줄이면서도 원 영상과 거의 유사한 영상을 복원할 수 있도록 해주기 때문에, 요즘 디지털 카메라 및 휴대폰 등 영상을 저장할 때 대부분 Exif(Exchangeable image file format)로 JPEG 압축형식을 널리 사용하고 있다. 본 논문은 JPEG 베이스라인 모드로 압축되어진 영상의 디코딩 단계에서 필요한 비계층형 헤더를 파싱하는 모듈의 기능을 소프트웨어로 모델링하고 VHDL을 이용하여 회로를 합성하고 동작을 검증하였다. 설계 결과 Xilinx xc3s1000 fg676 -4 환경에서 154.488㎒의 동작속도를 나타내었고, JPEG 디코더의 고속 데이터 처리에 적용가능하다.

Ⅰ. 서 론

JPEG 베이스라인 영상 압축 기법은 DCT(Discrete Cosine Transform) 기반의 영상 압축 기법이다[1]. DCT를 기반으로 하는 JPEG 압축 기법은 높은 압축률을 보일 뿐 아니라, 요즘 많은 종류의 이기종 디바이스에서도 JPEG 포맷을 서로 지원하기 때문에 사용자 대부분 JPEG 포맷을 많이 사용한다[2].

JPEG 베이스라인 모드로 압축된 영상에는 반드시 비계층형 헤더 정보가 포함되어 저장된다[1]. 헤더정보의 구성 및 데이터는 사용자 및 디바이스에 따라 서로 다를 수 있는데, 이것은 영상 인코딩단계 이전에 사용자에 의해 정해지는 이미지 사이즈, 압축률 등의 정보 및 여러 종류의 해당 디바이스에서 생성하는 다른 헤더정보가 포함되어 저장되기 때문이다. 그러므로 헤더 파서는 매번 다를 수 있는 헤더 데이터에서 필요한 정보를 추출할 수 있어야 하며, 휴대폰과 같은 다기능의 디바이스는 저전력 및 고속으로 동작하여야 하기 때문에 JPEG 코덱 코어뿐만 아니라 헤더 파서 모듈까지 하드웨어로 구현할 필요성이 있다.

본 논문에서는 가변적인 구조 및 데이터를 갖는 영상 헤더를 파싱하고 각 디코딩 단계에서 필요한 정보를 추출하는 헤더 파서의 구조를 연구하였다. 연구한 내용을 소프트웨어로 모델링 하였고 VHDL언어를 이용하여 하드웨어 모듈로 구현 및 검증 하였다.

Ⅱ. 파일 구조 및 헤더 파싱 흐름

그림 1과 같이 JPEG 베이스라인 모드로 압축된 파일의 구조는 크게 헤더 데이터와 압축된 이미지 데이터로 나뉜다[1]. 이 데이터들은 SOI(Start Of Image)와 EOI(End Of Image)마커코드 사이의 프레임 영역에 위치한다. 또한 양자화 테이블, 허프만 테이블, 이미지 정보등의 헤더 데이터는 서로 다른 마커 코드로 구분된 영역에 위치하며, 압축된 이미지 데이터는 MCU(Minimum Coded Unit)로 헤더데이터 뒤에 저장된다.

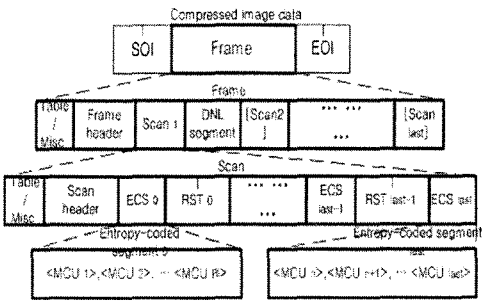

그림 1. 압축된 파일 구조

그림 2는 JPEG 이미지 디코딩 수행을 위한 헤더 파서의 전체적인 동작의 흐름 및 추출되는 헤더 데이터를 나타낸다. JPEG 이미지가 입력되면 헤더 데이터가 JPEG 헤더 파서에 입력되고, 입력

데이터가 파싱 가능한 유효한 마커 코드를 갖는지 판단한다. 유효한 마커 코드일 경우, 각 해당 루틴으로 점프해 파싱을 수행하며 이때 JPEG 이미지 디코딩 단계에서 필요한 여러 가지 정보를 추출하고 그 데이터를 RAM이나 Register에 저장한다.

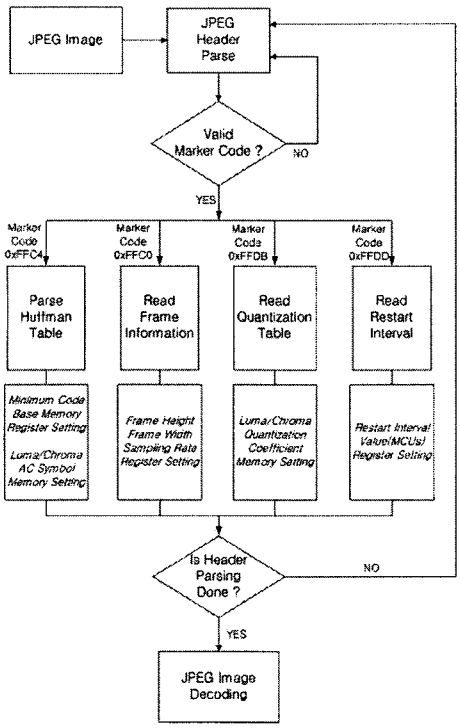

그림 2. 헤더 파싱 흐름도

이러한 동작은 여러 가지 헤더의 파싱이 완료될 때까지 수행되며, 완료가 되면 JPEG 이미지 디코딩 단계가 수행된다.

Ⅲ. 모듈 구현

그림 3은 설계한 모듈의 신호 입출력을 나타낸다. reset신호가 셋팅되고 enable신호가 활성화 되면 동작을 수행한다. 순차적으로 8-bit의 헤더 데이터가 입력되고, 압축된 이미지 디코더에서 필요한 양자화 테이블, Minimum Code, Base Memory, 허프만 심볼 테이블, 이미지 사이즈 등의 데이터들이 출력되고, 동시에 해당 데이터가 저장될 RAM의 주소 데이터 및 Write Enable 제어신호도 같이 출력된다.

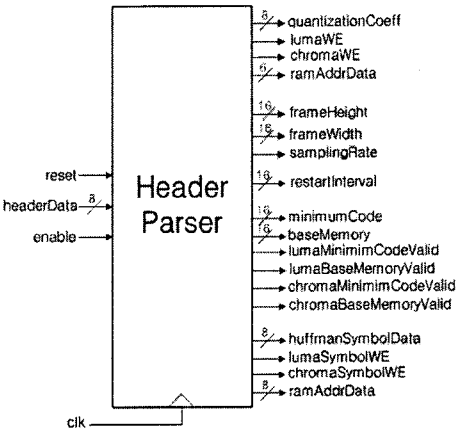

그림 3. 신호 입출력

각 헤더 정보를 파싱하는 동작의 상태들은 그림 4와 같다. 최초 Idle 상태에서 0xff가 감지되면 다음 클럭에 입력되는 마커 코드의 유효성을 확인하는 상태로 이동한다. 마커 코드가 유효하다면 해당 헤더 데이터를 파싱하는 상태로 이동하고, 그 상태의 동작이 완료되면 제어신호를 출력하고 다시 다음 클럭에 입력되는 마커 코드의 유효성을 확인하는 상태로 이동한다. 이때 마커 코드가 유효하지 않다면 Idle 상태로 이동하여 처음 동작을 반복 수행한다. 각 헤더 파싱의 상태가 완료될 때 출력되는 제어신호들을 매 클럭 체크하여 모든 헤더의 파싱완료가 감지되면, 중지상태로 이동하고 JPEG 이미지 디코더에 제어권을 넘겨준다.

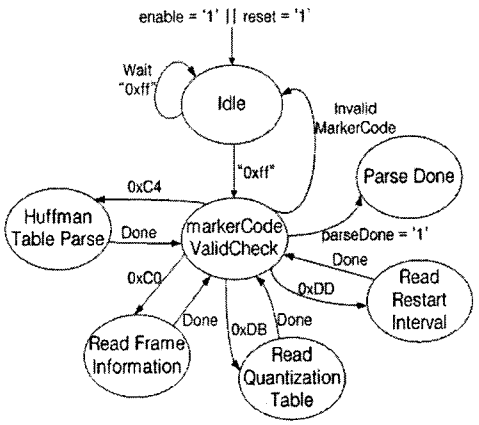

그림 4. 상태 머신

제어권을 넘겨주면 첫 단계로 VCD(Variable-Length Decoding)가 실행 되는데, Lei-Sun VLC(Variable-Length Coding) 디코더의 경우 PLA 혹은 ROM에 테이블을 저장하고 입력되는 코드워드와 직접 비교매핑 하여 심볼 값을 추출하는 구조로서, 산술 연산을 수행하여 동작하는 프로그램 가능한 VLC 디코더보

다 수행 속도가 느리다[3]. 따라서 프로그램 가능한 VLC 디코더를 사용하는데, 허프만 테이블을 파싱할 때 산술 연산되어 추출되는 Minimum Code 와 Base Memory는 입력되는 코드워드의 심볼값이 저장되어 있는 RAM의 주소 데이터를 계산할 때 사용된다[3].

$$C_k^{\min} = C_k^0 = \sum_{j=1}^{k-1} 2^{k-j} N_j \qquad [식\ 1]$$

여기서, k = 코드워드 길이 (1, 2, 3, ... , 16)
N_j = j길이를 갖는 코드워드 빈도수

식 1은 k 길이를 갖는 코드워드 중 가장 작은 값을 갖는 Minimum Code($C_k^{\min}$)를 유도하는 식이며, 표 1은 각 k의 Minimum Code 값이 추출되는 타임 테이블을 나타낸다. 이때, N_k는 입력되는 값이며, $C_k^{\min}$는 $C_{k-1}^{\min}$ 및 N_{k-1} 에 영향을 받는다.

표 1. Minimum Code 추출 타임 테이블 예

Time	k	N_k	$C_{k-1}^{\min}$	N_{k-1}	$C_k^{\min}$
T0	1	0	0	0	0
T1	2	2	0	0	0
T2	3	1	0	2	4
T3	4	3	4	1	10
T4	5	3	10	3	26
T5	6	2	26	3	58

식 2는 c_i에 대응하는 심볼값이 저장 되어있는 RAM의 주소를 계산하는 식이다[3]. 여기서 $[M(C_L^{\min}) - C_L^{\min}]$는 표 2의 Base Memory를 나타내며, $M(C_L^{\min})$는 이전까지 누적된 코드워드 빈도수와 같다.

$$M(c_i) = M(C_L^{\min}) + c_i - C_L^{\min} \qquad [식\ 2]$$
$$= c_i + [M(C_L^{\min}) - C_L^{\min}]$$

여기서, c_i = 입력되는 코드워드
L = 입력되는 코드워드의 길이
$C_L^{\min}$ = L의 Minimum Code
$M(C_L^{\min})$ = Base Address

표 2. Minimum Code와 Base Memory 테이블 예

k	N_k	Minimum Code Memory	$M(C_L^{\min})$	Base memory
2	2	0	B	B
3	1	4	B+2	B-2
4	3	10	B+3	B-7
5	3	26	B+6	B-20
6	2	58	B+9	B-49

식 3은 식 1, 2를 이용하여 하드웨어에 적합하게 Minimum Code 및 Base Memory를 계산하는 수식이다.

$$C_k^{\min} = (C_{k-1}^{\min} + N_{k-1}) << 1 \qquad [식\ 3]$$

k Base Memory = N_{k-1} ACC Value - $C_k^{\min}$

여기서, k = 코드워드 길이 (1, 2, 3, ... , 16)
N_k = k길이를 갖는 코드워드 빈도수

그림 5는 식 3을 이용하여 코드워드 길이별 빈도수 데이터를 입력으로, VLD의 심볼 추출 과정에서 필요한 Minimum Code 및 Base Memory 데이터를 연산하여 출력하는 블록도 이다.

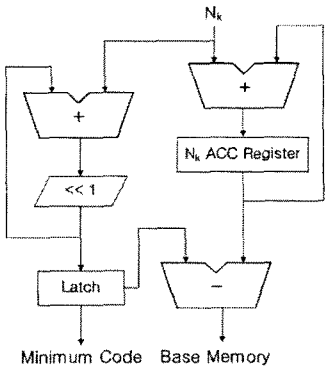

그림 5. MinimumCode 및 BaseMemory 연산 블록도

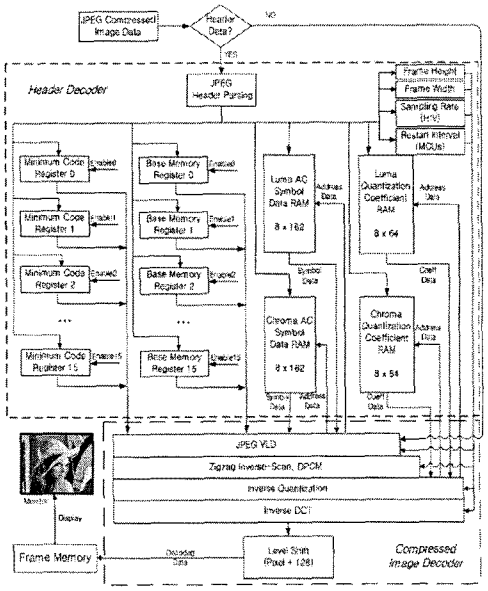

그림 6. JPEG 이미지 디코딩 흐름도

그림 6은 압축된 이미지 데이터가 입력되면 헤더파싱 모듈이 동작하여 Minimum Code, Base Memory, Frame Height, Frame Width, Sampling Rate, Restart Interval을 Register에 저장하며, Luma/Chroma Quantization Coefficient 8x64 RAM과 Luma/Chroma Huffman Symbol 8x162 RAM을 셋팅한다. 셋팅이 완료되면 추출된 헤더 데이터를 이용하여 압축된 이미지 디코더가 순차적으로 수행되며, 디코딩된 데이터는 Frame Memory에 저장되어 Display 된다[4].

Ⅳ. 결과 분석

그림 7은 설계된 모듈에서 양자화 테이블을 파싱하는 상태의 입출력 파형을 나타낸다. 양자화 테이블 마커코드 및 데이터가 입력되면 해당 상태머신이 동작하고 Luma/Chroma 양자화 계수 데이터 및 저장될 RAM 주소와 Write Enable 제어신호가 동시에 출력된다.

이때 전체 헤더 정보를 파싱하는 시간 및 순서는 입력되는 JPEG 이미지 마다 다르고, 표 3은 설계된 모듈의 FPGA 합성결과 및 특징을 나타낸다.

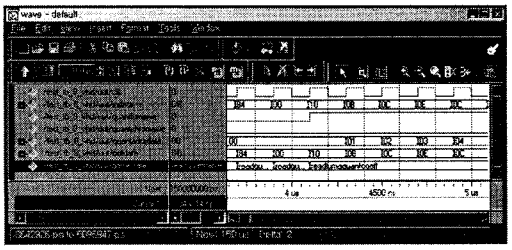

그림 7. 입출력 파형도

표 3. 설계된 모듈의 특징

No. of Slices	No. of Gates	Timing	Target Device
143	3,303	Minimum Period: 6.473ns Maximum Frequency: 154.488MHz	xc3s1000 -4fg676

V. 결 론

여러 디바이스에서 사실상의 정지영상 표준으로 JPEG을 널리 사용하고 있다. 또한 저전력 이면서 면적 효율적이고 고속으로 동작할 수 있도록 FPGA를 이용한 JPEG 코덱 구현기술이 계속적으로 연구 되고 있다.

본 논문은 JPEG 코덱 구현기술 중 헤더정보를 분류하고, 분류된 정보를 메모리에 저장하는 헤더 파서 모듈을 상태머신으로 설계하여 VHDL을 이용하여 하드웨어로 구현 및 합성 하였으며, 특히 JPEG 디코딩 단계 중 VLD는 입력되는 코드워드 데이터로 해당 심볼을 추출할 때 허프만 테이블 파싱 단계에서 셋팅되는 Minimum Code 및 Base Memory 데이터가 필요한데, 이 데이터들을 유도하는 수식을 하드웨어에 적합한 구조로 회로를 설계하여, 고속으로 해당 데이터를 연산하여 출력 하였다. 또한 설계된 모듈과 디코더 코어 사이의 효율적인 데이터 인터페이스 구조를 연구하였다.

VHDL을 이용하여 하드웨어로 구현된 JPEG 베이스라인 헤더 파서 모듈은 JPEG 디코더의 고속 데이터 처리에 적용가능하다.

참고문헌

[1] ITU-CCITT, Information Technology Digital Compression and Coding of Continuous-Tone Still Images Requirements and Guideline, CCITT, 1993.

[2] 정성태, "Visual C++를 이용한 실용 영상처리", 생능출판사, 2007.

[3] Vasudev Bhaskaran · Konstantinos Konstantinides, "IMAGE AND VIDEO COMPRESSION STANDARDS", Kluwer Academic Publishers, 1997.

[4] Mohammed Elbadri, Raymond Peterkin, voicu Groza, Dan Ionescu, and Abdulmotaleb El Saddik "HADWARE SURPPORT OF JPEG", IEEE, 1995.