FPGA를 이용한 JPEG 압축모듈 구현

*위지호 **유현 ***유승원 정제창

* **한양대학교 융합전자공학부 ***한양대학교 전자컴퓨터통신공학과

*wlwlfn@naver.com **ryoo202@naver.com ***ahddl1324@hanyang.ac.kr jjeong@hanyang.ac.kr

Implementation of JPEG Compression Module Using FPGA

*Jiho We **Hyun We ***Seungwon Yu Jechanh Jeong

* **Department of Electronic Engineering, Hanyang University

***Department of Electronics and Computer Engineering, Hanyang University

요약

정보를 전달하는 데에는 여러 수단이 있다. 그 중 가장 많은 양의 정보를 가장 직관적으로 전달하는 수단은 영상이다. 그렇기 때문에 사람들은 예로부터 공연과 연극을 통해 시각적인 형태로 내용과 분위기 등 다양한 정보들을 전달해왔고, 오늘날에는 영화 필름의 형태로 전달하기도 한다. 현대에는 기술이 점점 발전하고 한 매체가 담을 수 있는 용량이 커지면서 통용되는 데이터량이 늘어나고 있고, 또한 개개인의 삶이 하나의 컨텐츠가 되면서 사회 전체적으로 유통되는 데이터량이 급증하고 있는 추세이다. 이러한 사회적 추이를 따라 영상을 효과적으로 운용하는 중요성이 대두되고 있고, 이를 위해 그 데이터량을 효과적으로 관리하는 다양한 압축 기술에 대한 연구가 이루어지고 있다.

최근 국내에서 5G 통신 기술이 상용화되는 것을 보면서도 알 수 있듯이 기술이 발전함에 따라 처리해야하는 데이터량이 많이졌고 많은 데이터량을 처리하는 속도도 발전하였지만 많은 데이터량을 효율적으로 줄이는 방법도 매우 중요해졌다. 이 점에 착안하여 본 논문은 JPEG 인코더를 구현해봄으로써 영상의 효율적인 압축에 대한 이해도를 높이고 압축 자체에 대한 개념을 깊이 있게 함양하기 위해 본 연구 주제를 선정하였다.

1. 서론

영상을 압축하는 방식에는 크게 두 가지 종류가 있다. 무손실 압축과 손실 압축인데, 손실압축의 경우 무손실 압축을 포함하고 있어서 압축률이 대단히 높아진다는 장점이 있는 반면, 데이터의 손실이 발생한다는 단점이 있다. 다행히도 발생하는 손실이 육안으로 드러나지는 않는다는 점에서 손실압축을 적절하게 적용하는 것은 영상자료를 관리하는 데 큰 기여를 하게 될 것이다.

무손실 압축은 화면에 반복적으로 등장하여 빈도가 높은 자료에 더 작은 비트를 할당하여 상대적으로 더 적은 빈도의 자료에 더 큰 비트를 할당하더라도 총 데이터량이 적어지게 만드는 원리를 이용하고 있다. 대표적인 예시로 허프만 코드 할당과 Run-length 코드 할당 방식이 있다. JPEG와 같이 연속된 세 자리 이상의 0이 자주 등장하는 경우 허프만 방식보다 Run-length 방식이 더 압축률이 좋다.

손실 압축은 무손실 압축에서 한 단계 더 나아가 더 높은 압축률을 위해 고안되었다. 손실이 있되 그 정도가 아주 작아서 사람이 알아챌 수 없는 정도의 손실이라면 그 손실을 감안하는 방식이다. 대표적인 예시가 우리에게 익숙한 MPEG, JPEG이다. JPEG의 경우 최대 20 : 1 수준으로 용량을 줄일 수 있다.

JPEG 압축 과정을 간략히 정리하면 Blocking, DCT, Quantization, Zig-zag Scanning, Entropy Coding 과정으로 나눌 수 있다. 먼저 Blocking 과정은 이후 과정인 DCT를 적용할 때 FDCT로 적용할 수 있도록 특정 조건을 만족시켜 그 속도를 빠르게 하기 위한 단계이다. FDCT 과정에서는 이 블록들을 재정렬하여 행렬곱의 형태로 변환을 취하게 된다. Quantization 부분에서는 FDCT를 통해 얻은 행렬의 의미 있는 데이터는 그대로 두고 0과 가까운 값들을 0으로 바꿔주는 작업을 수행한다. 이 과정에서 데이터 압축과 손실이 발생한다. Zig-zag Scanning 과정에서는 좌측 상단에 집중된 데이터들의 정보들을 효과적으로 읽어들이는 과정이다. 데이터의 형태에 따라 수직 방향, 또는 수평 방향으로 읽는 것이 더 효과적일 수 있고, 다른 제원에서 이를 지원하기도 하지만 JPEG에서는 Zig-zag만 지원하고 있다. 마지막으로 Entropy Coding 과정에서 앞서 언급한 Run Length Coding을 적용하여 빈도에 따라 자료들에게 이진 데이터의 크기를 차등 할당하는 작업을 진행하였다. 이러한 과정을 통해 Encoding을 진행한 후 다시 역순으로 Decoding을 수행하여 최종적으로 압축된 JPEG 이미지를 얻을 수 있다.

이러한 JPEG 압축모듈을 구현하기 위해 FPGA(Field Programmable Gate Array)를 사용하였다. FPGA는 일반적인 반도체(ASIC)에 비해 느리고, 복잡한 것이 어려우며, 소비 전력이 크다는 단점이 있지만, 현장에서 즉시 오류를 수정할 수 있고 초기 개발에 용이하여 간단하고 실험적인 연구를 수행하기에 적합하다. 그리고 FPGA의 동작을 정의하기 위해 Verilog를 사용하였다.

본 연구를 통해 학교에서 배운 JPEG의 원리를 실질적으로 구현해봄으로써 영상 압축 기술에 대한 이해도를 높이는 데 큰 도움이 되었다.

2. 본론

JPEG 압축 모듈을 FPGA에 구현하기 위해 Verilog를 이용해서 코드를 짰고, 이를 위해 Modelsim 프로그램을 활용하였다. FPGA 모델은 Altera DE1-SoC를 사용하였다.

아래 사진은 JPEG 압축 과정에 대한 간단한 블록도이다. I_1은 모듈에

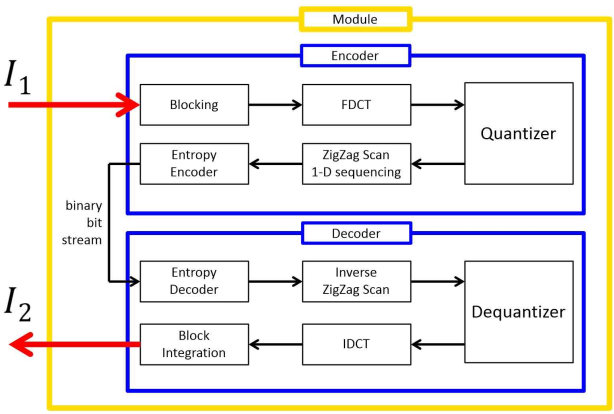

그림 1. JPEG 압축모듈 블록도

들어가는 이미지이고, I_2는 모듈에서 출력되는 이미지이다.

블록도의 각 항목에 대해 하나씩 아래와 같이 설명한다. Decoding 과정은 Encoding 과정을 역순으로 진행하여 순서대로 풀어내는 과정이므로 Encoding의 각 항목을 설명하면서 Decoding의 경우까지 포함하여 설명한다.

1) 8x8로 Blocking하는 과정

JPEG 영상 압축 중 DCT 과정에 FDCT를 적용하기 위해 가로 세로 모두 2의 N제곱 꼴로 표현 가능한 수를 정하여 block으로 분할하고자 하였고, 임의로 8*8 형태의 block을 차용하였다. 추가적으로 block 단위로 DCT를 적용하는 경우, 전체에 DCT를 적용하는 경우에 비해서 block 내 데이터값이 균일해지는 지점이 있었고, 이 지점에 맞게 blocking을 실시한 후 DCT를 취할 때 균일한 데이터로 인해 작업속도가 빨라지는 이점이 있었다. blocking을 구현할 때는 특정 객체를 생성하기보다 코드를 8개 단위로 끊어서 적용하는 것으로 구현하였다. Decoding 때에는 실제 값들을 블록 단위로 이어붙이는 경로를 지정함으로써 구현하였다.

2) FDCT 과정

DCT는 저주파대역의 값을 가지는 부분이 적게 나오고 고주파 대역에서 0값이 많이 나오기 때문에 압축률을 높이기에 용이하다. 이에 따라 본 연구팀은 해당 연구에서 DCT를 이용하여 JPEG를 구현하고자 하였다.

DCT 수식은 아래와 같다.

$$X[k] = \sum_{n=0}^{N-1} x[n] \cos \frac{\pi}{N}(n+\frac{1}{2})k \qquad (1)$$

DCT를 적용할 값들을 순차적으로 읽어들이는 작업과 옮겨쓰는 작업을 병행할 수 없고, 행과 열의 값을 하나의 모듈로 얻어내기 위해서 전치행렬을 사용하였고, 행과 열에 대해 각각 모듈을 적용하였다. Decoding 과정에서도 같은 방식을 사용하여 반대로 적용하였다.

3) Quantization 과정

Quantization Table							
16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

표 1. JPEG 표준 Quantization Table

임의의 quantization table을 활용하여 0에 가까운 데이터를 모두 지워냈다. 사용한 table은 아래와 같다.

Decoding 과정에서는 Table의 역행렬을 취하여 연산하였다.

4) Zigzag scanning 과정

지그재그 순으로 센서를 움직이도록 구현하는 함수가 매우 복잡하여, 지그재그로 움직이게 되는 실제 경로를 일일이 지정하였다. Decoding 과정에서도 지그재그로 읽을 때 순서대로 출력될 수 있도록 위치를 일일이 지정하였다.

5) Entropy Coding(Run length coding) 과정

Run length coding을 pre 모듈과 stack 모듈 두 가지로 나누어서 구현하였다. pre 모듈에서는 block 내 각각의 줄에서 part 모듈로 Encoding 값을 도출하였고 stack에서 이를 이어주는 방식을 채택하였다. Decoding 과정에서는 할당한 부호를 해체하지 않고 인식하는 과정으로 적용하였다.

3. 실험결과

위와 같이 verilog 코드를 완성한 후 PC에 FPGA를 연결하여 컴파일하여 FPGA의 동작을 정의하여 모듈을 완성하였다. 원본 이미지와 모듈을 이용하여 압축한 이미지는 다음과 같다.

(a)

(b)


(c)


(d)


(e)


(f)


(g)


(h)


(i)


(j)

그림 2. (a), (c), (e), (g), (i) 압축 전 원본 Image. (b), (d), (f), (h), (j) 자체 제작한 모듈로 출력한 Image. 표2에서는 각각 순서대로 Image #1~5로 표기되어 있다.

이 PSNR 값을 실제 JPEG 압축과 비교해 보았다.

PSNR [dB]	고안한 모듈	실제 JPEG
image #1	33.97	37.80
image #2	35.27	78.23
image #3	33.23	37.17
image #4	32.06	37.28
image #5	34.06	36.45
평균값	33.72	45.39

표 2. Output Image들의 PSNR값 비교

압축률 또한 아래와 같이 비교해 보았다.

압축률	사진 용량	고안한 모듈		실제 JPEG	
		결과용량	압축률	결과용량	압축률
image #1	262kb	64kb	75.5%	31.8kb	87.9%
image #2	257kb	64kb	75.0%	26.4kb	89.7%
image #3	257kb	64kb	75.0%	36.6kb	85.8%
image #4	257kb	64kb	75.0%	35.3kb	86.3%
image #5	262kb	64kb	75.5%	33.3kb	87.3%
평균값	259kb	64kb	75.2%	32.7kb	87.4%

표 3. Output Image들의 압축률 비교

4. 결론

실험 결과, 자체적으로 구현한 모듈로 얻은 결과 Image의 PSNR값이 실제 JPEG 압축 결과 Image의 PSNR값에 비해 근소하게 부족하였다. 그러나 32dB 이상의 PSNR값을 가지는 사진은 육안으로 품질의 차이를 인지할 수 없다. 자체적으로 구현한 모듈로 얻은 결과 Image의 PSNR 값이 32dB 이상이었으므로, 자체 구현한 해당 모듈 또한 품질 저하 없이 압축이 가능한 모듈이다.

또한, 손실압축의 특성상 원본 Image에 비해 데이터량이 다소 감소하였지만, Decoding 과정을 통해 Image를 복구할 수 있었다. 따라서 해당 모듈을 신뢰할 수 있는 압축 모듈로 간주할 수 있다.

용량 [kb]	원본 Image	복구 Image	손실율
Image #1	268.58	243.93	9.2%
Image #2	263.91	239.98	9.1%
Image #3	263.91	249.42	5.5%

Image #4	263.91	204.94	22.3%
Image #5	268.90	244.80	9.0%

표 4. Output Image들의 Decoding 손실율

표준 JPEG 압축 과정 중 Entropy Coding 단계에서는 Run Length Coding과 함께 Huffman Coding이 중복 적용된다. 해당 모듈에서는 Huffman Coding을 구현하지 않아 압축률이 표준 JPEG 압축보다 낮은 것으로 분석하였다.
모듈을 구상하고 구현하며 DCT, Quantization Table, Entropy Coding 등의 과정에서 보다 더 우수한 PSNR과 압축률을 구현할 수 있는 여지가 있음을 확인하였다.

참고문헌

[1] Alan V. Oppenheim, Ronald W. Schafer , *Discrete time signal processing,* 3th ed, Pearson Education, 2007.

[2] Rafael C. Gonzalez, and Richard E. Woods, *Digital image processing*, 4th ed, Pearson Higher Education, 2018.

[3] 최지은, 'JPEG 영상 압축에서 양자화 잡음 모델링,' *석사학위논문, 이화여자대학교,* 서울, 1999.

[4] 부선영, 'An Adaptive Generation of Quantization Tables in JPEG Image Compression,' *박사학위논문, 이화여자대학교,* 서울 2001.

[5] C. Yang, W. M. Lee, and L .W. Chang 'Designing JPEG quantization tables based on human visual system,' *Elsevier Signal processing : image communication,* Vol 16 issue 5, pp. 501-506, 2001.